

AFRL-PR-WP-TR-2002-2013

**THERMAL SYSTEM ANALYSIS
TOOLS (TSAT)**

**Ernest S. Hodge
Marvin R. Glickstein**

**MODELOGICS, INC.
4228 CARILLON TRACE
KENNESAW, GA 30144**



JANUARY 2002

FINAL REPORT FOR 01 SEPTEMBER 1999 – 31 DECEMBER 2001

Approved for public release; distribution is unlimited.


**PROPULSION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7251**


NOTICE

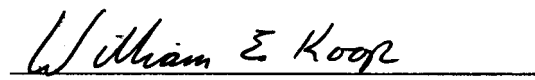
Using government drawings, specifications, or other data included in this document for any purpose other than government procurement does not in any way obligate the U.S. Government. The fact that the government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey and rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report has been reviewed by the Office of Public Affairs (ASC/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


VALERIE J. VAN GRIETHUYSEN
Engine Integration & Assessment Branch
Turbine Engine Division
Propulsion Directorate


JEFFREY M. STRICKER
Chief, Integration & Assessment Branch
Turbine Engine Division
Propulsion Directorate


WILLIAM E. KOOP
Chief of Technology
Turbine Engine Division
Propulsion Directorate

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YY) January 2002		2. REPORT TYPE Final		3. DATES COVERED (From - To) 09/01/1999 – 12/31/2001	
4. TITLE AND SUBTITLE THERMAL SYSTEM ANALYSIS TOOLS (TSAT)				5a. CONTRACT NUMBER F33615-99-2-2919	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER 62203F	
6. AUTHOR(S) Ernest S. Hodge Marvin R. Glickstein				5d. PROJECT NUMBER 3048	
				5e. TASK NUMBER 05	
				5f. WORK UNIT NUMBER EY	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Modelogics, Inc. 4228 Carillon Trace Kennesaw, GA 30144				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Propulsion Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson Air Force Base, OH 45433-7251				10. SPONSORING/MONITORING AGENCY ACRONYM(S) AFRL/PRTA	
				11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-PR-WP-TR-2002-2013	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES Report contains color.					
14. ABSTRACT <p>Design of military aircraft, or other complex engineering systems, involves designing the individual components and subsystems, and subsequently integrating all of those parts to yield the desired final product. Since the design of a new product (e.g., aircraft, spacecraft, etc.) usually begins with a definition of what the role or mission of the new product will be, the design process can focus on satisfying that mission. This design process, to be efficient, optimizes the total integrated design to best attain its product goal. This process has historically been a lengthy iterative procedure, often not arriving at an optimized, or even a satisfactory design.</p> <p>A new design approach is introduced and developed in this program for design of highly integrated systems. Focusing on one aspect of the total (aircraft) system, namely the thermal management systems, a new method for design, integration, and analytical evaluation is developed and demonstrated. This process allows modeling and simulation of highly integrated complex systems, and top level evaluation of such systems, thereby identifying the effects of subsystem design on the performance of the whole. The new methods are based on the use of object oriented programming methods, with open architect programming methods consistent with the Microsoft COM protocols. The resulting analysis methods are compatible with planned computer industry initiatives for distributed computing, using desktop computing platforms, and offer very powerful new methods for integrated design processes.</p>					
15. SUBJECT TERMS Thermal Management, Thermal System, System Integration, Modeling, Simulation					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor) Valerie J. Van Griethuysen 19b. TELEPHONE NUMBER (Include Area Code) (937) 255-4885
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
FOREWORD	v
1.0 SUMMARY	1
2.0 INTRODUCTION.....	3
3.0 Methods, Assumptions, and Procedures	7
3.1 Methods.....	7
3.2 Assumptions	9
3.2.1 Selected Components	12
3.3 Procedures	13
3.3.1 Development of System Utilities	13
3.3.2 Thermal Library Development.....	14
3.3.3 WinTherm/RadTherm Interface	15
4.0 Results	18
4.1 The TSAT Toolbox Additions	18
4.1.1 Heat Exchangers:	20
4.1.2 Turbomachinery:	21
4.1.3 Fluid Flow Components:.....	21
4.1.4 General Engineering Utilities:.....	22
4.2 Model Engineer System Utilities	24
4.2.1 Connectors:	24
4.2.2 Controllers:.....	24
4.2.3 Data Generator:	24
4.2.4 Schematic Viewer:	24
4.2.5 ReadXcells and WriteXcells	24
4.2.6 DoLoop	25
4.2.7 PasteView ToPPT	25
4.2.8 EditProperties.....	25
4.2.9 Log Watch.....	25
4.3 Object Engineer.....	25
5.0 Application and Discussion of Program Results.....	26
5.1 Component Object Behavior.....	26
5.2 Integration of Multiple Components	30
5.3 Integration of a Complex System.....	34
5.4 Development of a System SuperObject	38
6.0 Conclusions	39
7.0 References	40
8.0 LIST OF ACRONYMS	41

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 1 – Serial Engineering Process	4
Figure 2. Development of the Design Process Cycle.....	5
Figure 3. Concurrent Engineering Process.....	6
Figure 4. Conceptual Thermal Management System for Hypersonic Aircraft	10
Figure 5. Arrangement of Traditional Federated Subsystem Integration	11
Figure 6. Integrated Subsystem Technology Concept	11
Figure 7. Interface Definition for Procedural Heat Exchanger Model.....	14
Figure 8. Interface Definition for DLL Object Model of Heat Exchanger	14
Figure 9. Cooled Burner Panel Used as Coupling Problem with WinTherm.....	16
Figure 10. Output Display from RadTherm for Analysis of Heated Panel.....	17
Figure 11. Visual Basic Development Window	19
Figure 12. Visual Basic Toolbox with TSAT Engineering Tools Added	19
Figure 13. Counterflow Hex Object Dropped on Design Form.....	26
Figure 14. General Property Page for Counterflow HEX06v3	27
Figure 15. Heat Exchanger Property Page Shows External Dimensions	28
Figure 16. Property Page Showing Hot-Side Properties	29
Figure 17. HxPerf Page Shows Performance Data	30
Figure 18. Simple Integration of Two Heat Exchangers.....	31
Figure 19. Execution Order is Defined in Data Generator.....	31
Figure 20. Performance of First Heat Exchanger, Cooled With Air.....	32
Figure 21. Performance of Second Heat Exchanger, Cooled with Fuel	32
Figure 22. Property Page for Connector	33
Figure 23. Architecture for Vehicle-Integrated Thermal Control System.....	34
Figure 24. TSAT Model of a Vehicle Integrated Thermal Control System.....	35
Figure 25. Execution Order of Integrated Thermal Model	36
Figure 26. Schematic Viewer Displaying Subsonic Cruise Performance Data	37
Figure 27. SuperObject Model of a Cryogenic Superconducting Power Supply	38

FOREWORD

This report was prepared by Modelogics, Inc. as part of an Air Force Dual Use Science and Technology program with the Propulsion Directorate, Air Force Research Laboratory, Wright-Patterson AFB, Ohio 45433-7251. The report describes work done under Contract F33615-99-2-2919, with administration and technical guidance provided by Ms. Valerie J. Van Griethuysen (AFRL/PRTA).

The Program Manager at Modelogics for this contract is Mr. Ernest S. Hodge. Technical engineering and programming support was provided by Dr. Marvin R. Glickstein for Pratt & Whitney (P&W), and by Mr. Keith Johnson and Dr. Leonard Rodriguez for ThermoAnalytics.

1.0 SUMMARY

This document is the final technical report of the Thermal Systems Analysis Tool (TSAT) program, conducted by Modelogics, Inc., in collaboration with ThermoAnalytics and P&W. A thermal system is any physical system in which thermal energy is generated, transferred, or stored. Such a system can be as simple as heat transfer from a hot fluid to a colder fluid, or as complex as the thermal control of a hypersonic aircraft. The need for thermal and energy control arises in most engineering systems (e.g., power generation and transmission systems, electrical equipment, environmental control equipment, etc.), and may be dictated either by requirements for component or structural survival, or by demands to achieve required component performance. As examples, the turbine in an aircraft engine is cooled to provide structural survival and desired life, electronic equipment components (avionics) are cooled to provide life and component performance, and cabin environments are controlled to provide habitable conditions for people. In advanced military aircraft, the extraordinary demands for thermal control gives rise to complex thermal systems, which may be highly integrated with the propulsion and power systems. As a result, the various subsystems (e.g., engine, environmental control system (ECS), thermal management systems, etc.) may be interactive, greatly complicating the design of the aircraft systems.

As a result of several aircraft studies in recent years, a concept for high level integration of air vehicle systems has evolved, culminating in the development and demonstration of the Joint Strike Fighter /Integrated Subsystem Technology (J/IST) Demonstrator.¹ From these efforts, it has become clear that high levels of integration can provide synergistic benefits to advanced aircraft systems, particularly those with heavy demands on thermal and energy resources. To effectively capitalize on these potential benefits, integrated system parametric trade studies and designs must be performed at the highest level, allowing optimization of the overall flight system architecture. The conventional methods previously used for design of thermal systems allow design of individual small subsystems, but have not provided capability for total system integration. Computational methods have not been available for large-scale integration of complex systems, except by the use of a few industry proprietary and limited modeling environments. This lack of design methodology, for use with integrated thermal control systems, was the motivation for initiating this TSAT program. The program objective was nominally to develop methods to provide capability for computer modeling and analysis of complex thermal systems in an open object oriented environment. The methods developed in the program should be user-friendly and should allow integration of the modeled thermal system into the larger integrated flight system model.

The approach selected to develop the modeling methods is based on the presumption of operating in the Microsoft Windows environment, and the selected methods make extensive use of the Windows protocol for data exchange, the Component Object Model, or COM.² Within this protocol, physical or abstract components are modeled as objects, and these objects follow the rules established by Microsoft as ActiveX[®] controls.² An ActiveX control is an executable program that can be represented by an icon in a graphical environment and will execute when the icon is dropped into an appropriate container (i.e., a Microsoft-defined graphical environment) or form. Programmed objects that follow these established rules are called COM-compliant, and can communicate with other similar objects or programs, whether on the same computer or remotely across networks. This selected approach provides an existing framework in which to develop simulations of systems with integrated components, either with existing components or with those developed by the user. Modelogics has developed, prior to the initiation of the TSAT program, a set of COM-compliant software tools for use in the Windows environment to aid in the development of object-based components or integrated models. These tools collectively form a package called Model Engineer (ME), providing the infrastructure and basic utilities for building and solving general system problems. These utilities allow the user to build simulations of complex physical systems without the need to write any program code. They provide mechanisms for connecting components and passing data, controlling the execution sequence in the system, controlling iteration loops, and providing mechanisms for data input and output. In addition, these utilities include a specialized code generation tool called Object Engineer (OE), which is specifically designed to aid users in building their own components.

Under the funding of the TSAT program, an extensive effort was made to provide an engineering library of component objects, representing the range of physical components needed to build typical thermal management systems projected for use in future high performance flight systems. These components include a wide array of heat exchangers, pumps, compressors, turbines, control valves, flow splitters and mixers, and other miscellaneous physical devices. In addition, a library of fluid properties was developed to provide thermophysical properties of a wide range of fluids, with the capability of adding new fluids as needed. The resulting modeling system, including ME and the new engineering libraries, allows building and analyzing complex system models in a friendly graphical environment. The initial goal of this program was to develop the desired modeling system with the combined capabilities of several existing systems, while providing the growth and distributed modeling capabilities inherent in the Windows environment. Specifically, it was desired to include the functionality available in the Vehicle Integrated Thermal Management Analysis Code (VITMAC),³ previously developed under Air Force sponsorship. The VITMAC package consists of a thermal management simulation code, integrated with a graphical user interface and a plotting package, for ease of output. The thermal management simulation operates in a proprietary environment, which does not provide easy addition of new user-defined components or capability for connectivity with remote simulations. By contrast, the ME/TSAT approach is based on object-oriented programming in an open architecture, allowing the user to develop and add new components as needed. This approach also provides capability for distributed modeling in a collaborative environment. Using an open architecture approach does not automatically mean that the methods or characteristics used in analyzing individual components are exposed to the world, but rather that the interface properties (inputs and outputs) are readily available for utilization. The details of a specific component object can be hidden, even though the object is made available for general analysis use. This allows the developer of a component to protect company proprietary information, while allowing users access to the component object for computation.

The new methods developed in this program have been verified by building and analyzing several complex aircraft systems, including a model of the J/IST demonstrator. An overview of the background that led to this program, the approach employed in developing the methods, and the accomplishments achieved were presented at the International Gas Turbine Institute as part of ASME Turbo Expo 2001, in New Orleans, LA.⁴

2.0 INTRODUCTION

The Problem

Modern aircraft, both military and commercial, incorporate a variety of component subsystems to provide propulsion, power, vehicle control, and thermal management. In high-performance military aircraft, the demands on these systems escalate, and resources for satisfying these demands become strained. Several of the subsystems have characteristics that are mutually synergistic, and the potential exists to better satisfy the on-board demands through high-level integration of the various subsystems, thereby capitalizing on this synergism. This possibility was recognized about 15 years ago, during design studies of conceptual hypersonic aircraft, and was later considered as a possibility for more conventional aircraft, leading to a series of study programs during the early to mid 1990's. These Air Force programs, the Subsystem Utility Integration Technology (SUIT),⁵ J/IST, and Vehicle Integration Technology Planning Study (VITPS)⁶ programs identified significant benefits that could accrue from high-level integration of vehicle systems, taking advantage of potential synergistic capabilities.

In the integrated systems examined to date, the primary thread that is common to the various subsystems has been the requirement for thermal control. This requirement appears in the propulsion system as the need for cooling of the hot section (e.g., turbine, disks, afterburner, and nozzle) and the lubrication system. In the vehicle, heat-generating components such as generators, avionics, and hydraulic systems must be cooled, and environmental control of the crew station must be provided. The thermal demands imposed by these different requirements depend on the technology level of the flight system components and the aircraft operating conditions. The required thermal conditions (heat load and allowable temperature) of the various components differ, but they all must share the available cooling resources. These resources, in a typical military aircraft, consist of the airflow in the propulsion stream (fan and compressor bleed air), the engine fuel, and additional ram air that may be brought on board specifically for cooling. To adequately satisfy the various thermal requirements, while minimizing aircraft performance penalties (i.e., aircraft weight, range capability), design of the various subsystems must be performed at the integrated vehicle level. This allows making best use of the available cooling resources, and provides an insight into the performance implications of selected integration architecture.

In a highly integrated system, interactions occur among many of the component subsystems, and it is desirable to understand how changes in individual components or subsystems affect the behavior of the entire system or vehicle. Optimization of such an integrated system can best be performed with simulation of the total system, thereby simultaneously capturing the effects of all the system interactions. This has been a difficult challenge because the individual subsystems are typically designed, modeled, and analyzed by each company or group responsible for that specific subsystem, and the various models are not mutually compatible. For example, in the design integration of a combat aircraft, the subsystems include the airframe, engines, ECS, electrical power generation and distribution components, avionics, and control actuation system. Design models of these various subsystems, and their component parts, are developed by the responsible contractors, and are treated as proprietary by each contractor.

Because the individual component models are not available, or are not compatible for integration at the system level, the integration process has conventionally been a series of long and drawn out iterations among the various subcontractors and component groups. As a result, optimization of the entire flight system has been minimal. The various component suppliers have been willing to share performance data for their components, but typically not willing to share their design methodology (i.e., computer models). Even if the component models were all available for system integration, providing mutual communication among all the disparate subsystems models is a monumental task. This becomes a greater issue in view of the fact that the system models must be re-done for each system architecture change or iteration.

Because of the current difficulties inherent in top-level optimization of complex integrated systems, and of the significant potential benefit of achieving such optimization, it has become apparent that providing better methods for system integration become a critical technology for designing future military flight

systems. This need for system integration tools defines the objective of the program being discussed within this report. Specifically, the objective of this program has been the development of methods and tools for the design and analysis of integrated thermal systems. Such systems include thermal management, environmental control, and their integration with the propulsion and fuel systems. Modeling environments have been developed in the past, for specific computer platforms, that provided capability for thermal modeling and analysis, but these were proprietary systems that were tailored to particular industries, types of problems or specific condition. Furthermore, these modeling environments did not provide capability for integration of subsystem models operating outside of the proprietary environment. Engineering, as practiced today in many organizations, is a serial process with a typical product development cycle like the one depicted in Figure 1. Typically, there are long feedback loops, often due to manual data interchange with compartmentalization of the information. With this approach, it is easy to see how subsystem groups tend to optimize designs for their box based on a set of parameters relating strictly to their discipline. They pass this design along and hope the restrictions they have implicitly put on all those subsystems downstream do not cause too much of a problem, i.e., necessity for redesign. There are several flaws with this process. First, due to the complexity of all the many aspects of the system design, it is almost impossible for anyone to completely understand everything, and because it takes so long to complete the design process, the product is never fully optimized. Only the individual parts of the design are optimized.

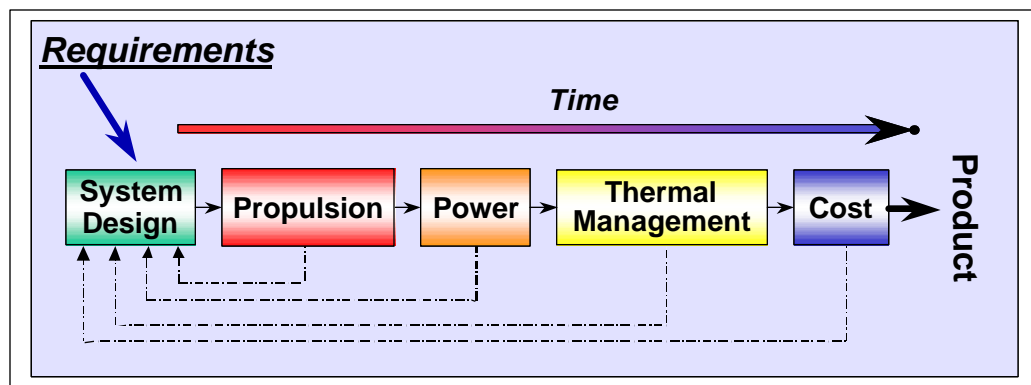


Figure 1 – Serial Engineering Process

Second, rarely are designers from different disciplines examining and running trade studies on identical versions of the system design, which of course has major implications for quality control. And third, once the design is completed it is impossible to repeat the process to determine the flow of requirements and the reasoning behind all of the original design decisions and constraints. This classic process results in a product that is not optimized for the overall requirements, but rather for the boxes within the design process. A suboptimized product design means either the product costs more than it should to produce or the performance or value of the product is low for the cost to produce. Either of these outcomes can mean millions of dollars of lost profit and excess costs to the corporation or government.

The engineering process as described above has been the same for decades. Originally, it was done with pencil, paper, and slide rule. Then in the 1970s people began to apply technology to the process, first in the form of calculators to speed up the mathematical calculations, followed by large, centralized computers. These in turn were followed by departmental large-frame computers running proprietary operating systems, and later in the UNIX-based workstation environment. The development of the design process over the years is illustrated in Figure 2.

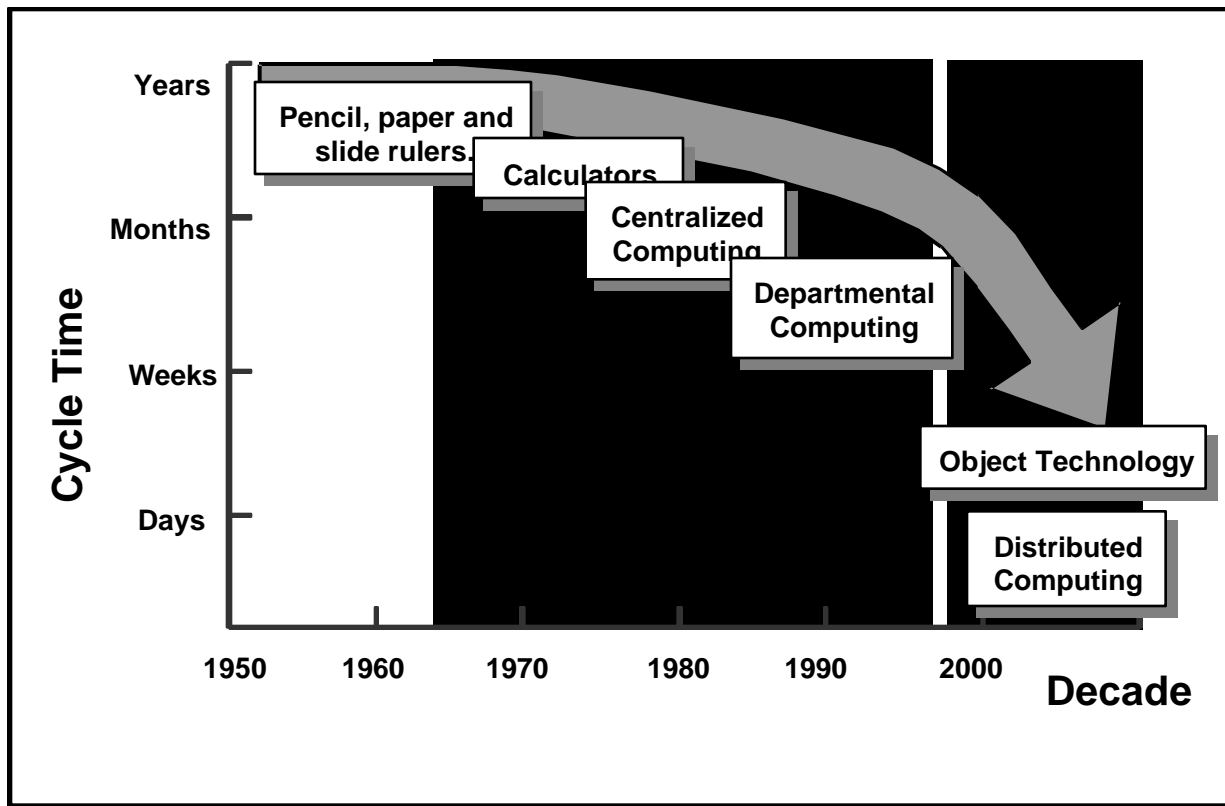


Figure 2. Development of the Design Process Cycle

All of these steps were simply automating certain aspects of the process, speeding up the component subprocesses and making incremental decreases in the design cycle time, but ultimately not making significant improvements in the overall process. Today, this approach has taken the form of trying to link the various steps in the engineering process to automatically transfer data from one computer program to the next. But such efforts have generally failed because the methods and systems that are trying to be linked are difficult to use individually, much less as a group. In many instances, linking of existing methods offers even less time savings than previous measures.

The reason for such an unexpected result is that, in using this approach of linking legacy programs, engineers spend most of their time trying to understand why the various programs failed to converge to a solution, and spend very little time actually examining results. Basically, automation of the current engineering process has hit a wall. The problem, simply stated, is that the commercial sector is demanding quicker time to market for new products, even mass customization... but the question is how to get there.

If there is to be revolutionary improvement in engineering approaches, then it is time to reengineer the whole developmental engineering process itself and bring together state-of-the-art computing technology and engineering. This is more than simply rewriting mainframe legacy codes or moving analysis codes from UNIX systems to the PC. It means the elimination of the old serial process where functional groups worked their portion of the effort and then passed along the results to the next group. In other words, a paradigm shift from the old tried and not so true approach to a more truly integrated, real-time and efficient approach or process makes sense.

A Solution

The new process must be one of concurrent engineering where all the functional groups are working on the same system version in parallel, immediately seeing the results that their design decisions have on the entire system. Traditionally, only a few project participants (project leaders, program managers, etc.) have been involved in the review of system level study results that also included cost, reliability, operability, and manufacturability. Now engineers will be involved with this review process. Further, engineers who have traditionally tried to optimize their discrete part of the problem will now be able to perform global optimizations on the entire system.

The new paradigm for engineering design and development unifies the computing platform by bringing all functions to the desktop in an integrated, easy-to-use environment, one that promotes concurrent engineering and focuses on system optimization instead of subprocess optimization (Figure 3). A virtual engineering environment that dramatically reduces the time to perform engineering trade studies, while making the engineer's job easier, is envisioned for this next generation engineering process. But more importantly, it will allow everyone from the various disciplines working on a project, to focus on their own expertise while simultaneously gaining an immediate big picture insight of the problem at hand.

The benefits of this new collaborative approach to the design and integration of complex engineering systems have been recognized for several years, and a variety of methods are evolving to take advantage of such benefits. These various approaches have been given several generic names, including collaborative engineering, distributed simulation, and concurrent engineering. Among these methods, the distinction between persons-in-the-integration-loop, and truly integrated software simulations has been somewhat fuzzy. Many software systems have been developed for simulation of complex engineering systems, and have provided adequate capability for component integration, as long as the targeted physical system fell within the planned scope of application. Most of these software integration methods have been based on proprietary simulation systems, since their development was prior to the open standards for interoperability that have evolved just during the last few years. These simulation systems become inadequate when the total scope of the required integration exceeds the scope of any of the individual software systems (i.e., integration of aircraft with power, thermal management, environmental control, electrical systems, etc.). In such cases of broad system integration, specific software is typically available for simulation of the different primary or secondary subsystems, but the software packages are not readily capable of communicating with each other.

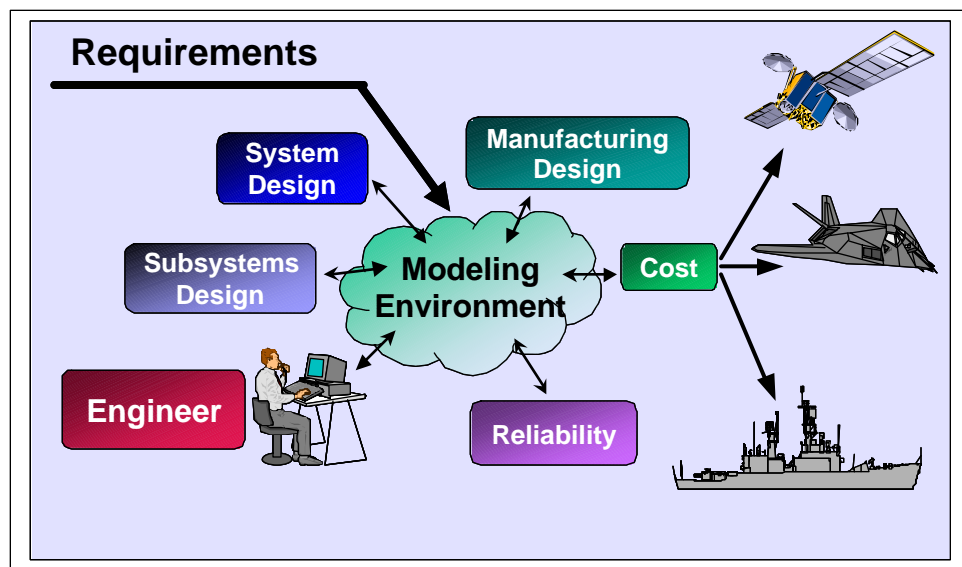


Figure 3. Concurrent Engineering Process

In the current program, to provide a more general method for modeling and analysis of thermal systems while providing a broad capability for system integration, an open (industry standard) environment is chosen in which to develop modeling methods and tools. This program is focused on developing methods for use in desktop computing in the PC Windows environment, adhering to the Microsoft protocol for distributed computing. This protocol, nominally the COM with ActiveX controls (i.e., objects) provides an environment in which objects (or applications) can readily pass information as needed. The same methods that allow passing data or images from Excel to PowerPoint allow passing data among objects that represent physical components in an engineering system. If physical components are modeled and programmed as ActiveX objects, compliant with the COM protocol, then they can be run in any COM-compliant Windows application, such as Excel, PowerPoint, Visual Basic, and Visio. Such component objects can then be assembled in an appropriate Windows environment to provide a simulation of a desired integrated system. The modeled system is then executed over the desired operating envelope to evaluate the performance of the integrated system.

Within the course of the program, the methodology has been defined and developed to simulate and analyze large-scale thermal systems, applicable primarily to advanced flight systems, yet with the flexibility to be used or readily adapted for other applications. The methodology takes advantage of the infrastructure developed by Microsoft for distributed computing in the Windows environment, as well as the utilities infrastructure developed by Modelogics within ME for data manipulation in the Microsoft COM environment. Under support of the Air Force TSAT program, a set of thermal components has been modeled as COM-compliant objects and made available as tools in a modeling library. These thermal components represent a group of physical devices that are used in the engineering design of thermal systems, both for power generation and thermal management. This component library, and the associated ME utilities, can be considered as a toolbox, with the component objects as tools. These tools can now be used in any appropriate Windows environment to model integrated systems. While the initial tools built under the TSAT program are applicable to thermal systems, additional components such as avionics, electronics, hydraulics and power generation systems, can be added to the toolbox library as needed to expand the capability of the modeling system. The resulting methodology provides a visual modeling environment, in which simulations are built by drag and drop methods, yielding a user-friendly approach to constructing top-level models of complex systems. The components comprising an integrated model may reside on a single computer, or may be remotely distributed across a network. Thus, individual component models may remain with their corporate owners, while communicating with an integrated system model across the network. This provides the capability for building high definition simulations of integrated flight systems in a collaborative environment, facilitating system optimization early in the design cycle. When implemented, this will provide an extraordinary advantage over current methods for design and integration of complex systems.

3.0 Methods, Assumptions, and Procedures

The objective of the TSAT program was to develop a software tool to simulate thermal management and control concepts for advanced propulsion technology, capable of performing trade studies on an entire system (e.g., aircraft, spacecraft, or weapon system). This product is to provide the Air Force and Department of Defense with the capability to quickly and independently assess thermal system concepts for existing, near-term, and far-term systems on a truly integrated basis. This task has now been completed within the defined scope of the TSAT program.

3.1 *Methods*

The beginning point for this task was the selection of the computing platform and environment for which the software would be developed, and the approach that would be taken in the design of the software system. The decision was made at the outset that the effort would be focused on desktop computing in the Microsoft Windows environment. That platform at the time was the primary platform for general office applications, and it appeared to be rapidly developing toward the capability of providing an extensive

environment for engineering applications. This choice has been substantiated by the rapid development of Intel (and compatible) based hardware during the period since the program began, and the projected path for development of this platform during the coming years. The impact of this platform choice on the commercial viability of the developed software is very clear. With the rapid acceleration in the capability of entry level desktop computers, and the shift for general engineering applications from UNIX to Windows platforms, the potential market for the software developed in this program is rapidly expanding.

Windows-based software applications operate with a graphical user interface, providing a visual and highly intuitive interface between user and application. However, there are other features that are available in these environments. Microsoft Windows provides capability for linking applications and objects, thereby passing data as needed. This process was originally called Object Linking and Embedding (OLE), and is now part of what is included in the COM.² Windows-based applications are not all COM-compliant, but can be designed to provide this capability, if desired. An application is COM-compliant if it obeys the relevant protocols defined by Microsoft for data passing. An object, as used in this document, is a thing (either tangible or intangible), defined by its name, and representing a real-world object. An object programmed as software has certain characteristics, including properties (or variables) and methods (or procedures). An object can execute as a stand-alone program, or as a component of a subsystem or larger system. A single component, such as a simple valve, can be modeled as an object, or a group of objects can be used to build an application, which in turn can become an object. If a physical object is modeled as a COM-compliant software object, it can be called from any application, just as any subroutine is called. Furthermore, such an object can be dragged and dropped into any COM environment (e.g., Microsoft Excel, PowerPoint, Visual Basic, Visio, Internet Explorer, etc.), where it can be exercised as a standalone program.

The approach selected to develop the desired software tool is based on assuring compliance with the Microsoft protocols for the PC/Windows platform. The basic program structure is object-oriented, and all objects are COM-compliant. The primary languages used in this TSAT program for object programming are Microsoft Visual Basic 6.0 (VB) and Digital/Compaq Visual Fortran 6.1 (Fortran). This version of Visual Fortran is Fortran 95 compliant, and is the latest version of the generic FORTRAN language. It is also compatible with the Microsoft Visual Studio development environment. All COM-compliant objects are programmed in VB, and are then compiled into ActiveX controls. These controls are COM objects, and are a special form of compiled code defined by Microsoft. They are installed in the computer, and are registered in the Windows registry, just like other applications. This is the form required of an object to allow drag-and-drop capability. Fortran is used in this program to model the more complex physical components, because of its design for scientific and engineering applications. Although such components could be modeled in VB, Fortran provides a more convenient programming environment for complex models. Fortran components can be compiled as objects in the form of dynamic link libraries (DLL), but these are not COM-compliant. To achieve this compliance, each Fortran component has an associated VB object as a front end, which provides compliance with the protocol, and passes data as necessary. The multilanguage programming used in this effort is available through the use of the Microsoft Visual Studio (V 6.0) development environment, which is specifically designed to allow such mixed language programming with VB, C, C++, and Fortran.

There are three general types of objects necessary to provide a modeling capability for simulation of thermal systems. These consist of the following:

- System utilities to provide operational management or control of the simulation
- Component models to represent physical engineering devices
- Engineering utilities to represent material and fluid properties, and general physical processes

Modelogics had completed a modeling infrastructure, prior to the start of the TSAT program, based on the use of the COM protocols. This infrastructure, called ME, provides system utilities to support further development of modeling methods. If a system simulation is considered as composed of various component objects, then the system utilities are those software devices that oversee the simulation, governing the sequence of operation, convergence of necessary control iterations, passing of data among

component objects, and capturing runtime error messages from the simulation. In addition, the system utilities provide methods for mapping data from the simulation to Excel for further use, and provide capability for presenting simulation output graphically during operation, or in a schematic viewer representing the physical system.

The tasks conducted under the TSAT program consisted of the development of a library of component models applicable for simulation of thermal control and power systems, and development of several engineering utilities. The engineering utilities provide thermophysical properties of fluids and structural materials, as needed for analysis of processes in the component models. In addition, several generalized processes were modeled as engineering utilities. The total group of component model objects and utilities form a set of tools, and in the Windows nomenclature, is added to the Controls Toolbox. This Toolbox includes a large collection of compiled objects, some developed by Microsoft, and others by third party developers, such as Modelogics. These objects are accessible to any application providing a COM-compliant environment. The tools developed in the TSAT program will be described in detail within this document.

3.2 Assumptions

The objective of this program was to build a tool for analysis of thermal systems. The platform environment and methods to be used for analysis were first selected, followed by identifying and defining the scope and types of thermal systems to be analyzed. Since the interface protocols and supporting system utilities were available from prior work, the primary focus of this program was the development of engineering component models and necessary engineering utilities. In addition, methodologies for the integration of the modeled components, and analysis of the resulting complex systems were addressed in the program.

Two initial presumptions were made regarding the scope of the program. First, it was decided that a thermal system, as considered in this program, would be defined as the integration of one or more discrete devices, such as pumps, valves, heat exchangers, compressors, turbines, or burners. The tools developed in this program do not apply to general continuum problems, such as generalized conduction, flow, and convection problems. Tools, such as computational fluid dynamics (CFD) methods, already exist for such problems. Discrete devices, such as a heat exchanger, could be modeled at whatever level of definition is desired, but the device is still defined as a specific physical component rather than a general fluid or solid field.

Second, it was decided that the modeling capability would be limited to steady-state systems for the current program. This does not preclude consideration of transient systems at a future time, since the methods used in this program can be readily extended for dynamic transient applications.

The next step in developing a component library was the selection of the components to be included in the TSAT toolbox. A committee consisting of the project team and the Air Force Program Manager did this selection. The selection process was conducted by examining several flight system applications that require thermal management systems, and identifying the various components necessary to build those systems. The applications considered include a hypersonic aircraft (Mach 5 class), a military fighter aircraft with a conventional ECS, and an advanced strike aircraft with integrated subsystem technology. The resulting component list from this exercise was then compared with the component library in the VITMAC component library, to ensure that no components from that library had been missed. In addition to physical components, supporting utilities were identified to provide physical properties of materials (structural and fluids), and common processes (e.g., gas dynamic processes). Fluid thermophysical properties would be available from a library, with initial fluids to include those commonly considered in power and cooling systems (e.g., air, fuels, fuel-air products, lubricants, refrigerants, etc.).

Figure 4 illustrates a conceptual thermal system arrangement for a hypersonic aircraft⁷ in the Mach 4-5 range that was considered in the component selection process. Cooling requirements include cockpit, avionics, flight controls, high speed (Ramjet) and low speed (turbojet) propulsion systems, and limited aircraft structure. In the system shown in Figure 4, the aircraft fuel is the primary heat sink at high speed, and all thermal control subsystems reject heat into the fuel stream. The integrated system must therefore manage the fuel flow as required by the engine, and incorporate thermal interfaces (heat exchangers) between the fuel stream and other secondary coolant streams. Components appearing in this application include fuel/air and fuel/secondary fluid heat exchangers (labeled as CHER), cooled structural panels, and

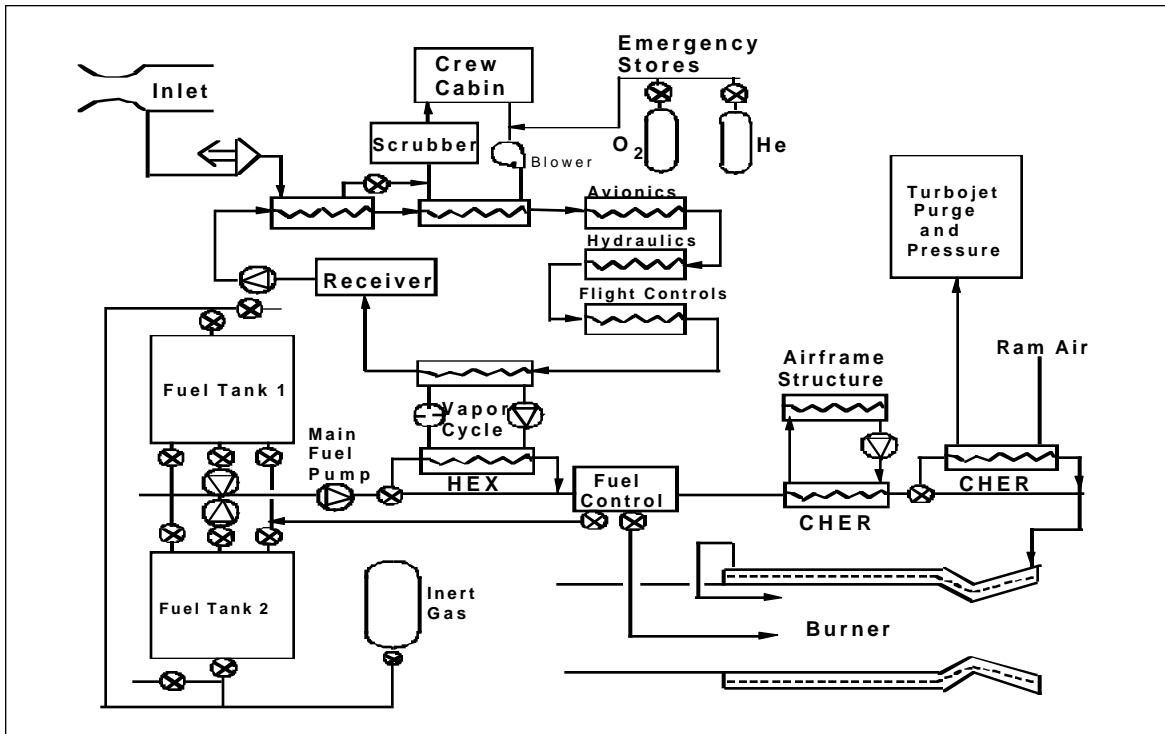
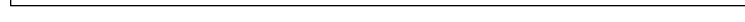


Figure 4. Conceptual Thermal Management System for Hypersonic Aircraft

pumps, compressors, and control valves. In addition, stream splitters, mixers, and other fluid components are seen as necessary when this system is examined in more detail.

Military strike/fighter aircraft have conventionally been designed with federated subsystems, in which the various subsystems (e.g., ECS, auxiliary power unit [APU], and emergency power unit [EPU]) have been independently designed and subsequently packaged into the air vehicle. A system integration of this type is shown as a schematic block diagram in Figure 5. In most aircraft, the ECS have been based on reverse Brayton cycles, with ram air used as the primary heat sink. In this concept, high-pressure bleed air from the engine compressor is cooled with ram air, then further compressed, cooled, and expanded in an external turbine to produce cold air. This combination of components is sometimes called an air cycle machine, and is the basis for onboard refrigeration for most aircraft today. In this system, air-air heat exchangers (HEX) are used to cool high-pressure bleed air with low-pressure ram air, requiring HEX designs with very low-pressure drop.

Because air is the refrigerant in this cycle, condensation of water in the expansion turbine can be a problem at low altitude, requiring that water separators be incorporated in the system. This introduces the requirement for psychrometric properties, and separator components, as well as heat exchangers with condensation modeling.



3.2.1 Selected Components

Reviewing the illustrated aircraft systems in detail, a list of system components was compiled from the perspective of providing the capability to model any of the systems considered. The following lists those component models selected for development under the TSAT program.

Heat Exchangers:

- Compact (Sheet-Fin) Counterflow
- Compact (Sheet-Fin) Multipass Cross-flow
- Single-pass Cooled Panels
- Intercooler (Defined Heat Load)

Turbomachinery:

- Centrifugal Pump
- Compressor
- Fan
- Turbine

Fluid Flow Components:

- Duct
- Control Valve
- Throttle valve
- Flow Splitter (Controlled)
- Multibranch Splitter
- Recirculating Splitter
- Flow Divider
- Mixer
- Multibranch Mixer
- Ejector-Mixer
- Flow Joiner
- Recirculating Flow Joiner
- Water Separator

General Engineering Utilities:

- Generalized Fluid Properties
- Generalized Material Properties
- One-Dimensional Gas Dynamics

Although several of these components appear to be similar, there are intrinsic differences that are best handled as separate components. These components will be further described in a later section of this report.

Prior to this program, a number of modeling systems had been developed for specific application areas, such as hypersonic aircraft thermal control and chemical process plants, based on proprietary or closed architecture environments. One such modeling system, the VITMAC, was developed specifically for thermal system analysis. While this code provides a library of thermal and flow components, and methodology for integrating these components into a system, it does not provide the interfacing capability implicit in open-architecture systems, such as COM. However, a stated objective of this program from the start was to develop a thermal system analysis tool with an open object-oriented architecture that initially provides the functionality of an engineering code such as VITMAC, and expand upon its capabilities. For this reason, the VITMAC component library was compared with the component list identified in this

program, to ensure that no significant physical components had been overlooked. The TSAT component list was found to include all of the VITMAC components, and was therefore accepted as adequate for the scope of the program.

3.3 Procedures

Each member of the project team had developed relevant modeling technology prior to initiation of the TSAT program, and these technologies form the basis for much of the resulting thermal analysis tool. Modelogics had established a modeling infrastructure to support object-oriented simulation in a COM-compliant environment, and had developed a set of component objects relevant to modeling vapor-compression refrigeration cycles. ThermoAnalytics had developed a set of COM-compliant two- and three-dimensional thermal analysis tools for evaluation of the thermal radiation characteristics of vehicles and related components. P&W had developed thermal simulation methodology for design and analysis of thermal management systems for advanced propulsion concepts. While the P&W methods were not based on object-oriented methods, they were based on modular methods, with physical components modeled as callable procedures or routines in a library. This allowed the subroutines in the component library to be readily converted to objects, without need for major additional programming.

The division of tasks among the team members was based on the prior technologies that each brought to the current program. Modelogics was responsible for continuing development of system utilities as requirements were identified during the program and for developing methods for incorporation of mixed-language programming into TSAT. P&W was responsible for the development of a thermal engineering library, compatible with and satisfying the requirements of the TSAT program. This would include conversion of Fortran legacy code for existing component models, and development of new components and engineering utilities, as required. ThermoAnalytics was responsible for developing and demonstrating an interface between the TSAT methods and their existing WinTherm (and RadTherm) software for thermal analysis and evaluation of thermal radiation signatures.

3.3.1 Development of System Utilities

The system utilities to be used in TSAT are based on those previously developed by Modelogics as part of their ME simulation toolset. Under the TSAT program, these tools were further enhanced, but will still be referred to as ME.

Many of the thermal library components to be added to the TSAT library are based on existing Fortran component models (originally written in FORTRAN 77), available from a previously developed procedural modeling system developed by P&W. This system, known as the Fuel and Integrated Thermal System Simulator (FITSS), maintained a library of components as callable subroutines. In the FITSS code, integrated systems are assembled in a main calling program, which calls the procedures for the various component models in the desired order, and also provides integration and convergence routines. The main program is then compiled and linked to the subroutines in the library to produce an executable program. In an integrated model, built with this approach, data is passed among components with named common blocks, providing public access to the various data elements from all components. This method of passing data is not allowed in object-based programming, where objects execute independently, passing data as necessary. Converting Fortran procedural subroutines to objects requires three major steps: 1) modifying the Fortran routines to a form in which data is only passed as a set of arguments, 2) compiling the Fortran module as a callable DLL object, and 3) building an object-based front end that can call the DLL object. The first two steps are described in subsection (3.3.2). The front end, written in VB, is a COM-compliant object that communicates with the other system components, and can call the specified component DLL. These objects can be compiled as ActiveX controls, and are graphical with drag-and-drop capability. It is possible to write a component object completely in VB, in which case no DLL is called, and the front end is the component object.

Because of the complexity of the code required for defining a graphical interface object, it was decided early in the program that a utility would be developed to simplify the task. This utility, named OE, automates the process of building a component object, either as a self-contained model, or as an interface for calling a DLL. OE is a utility that aids in the building of new components. If the object to be built is an interface, then the characteristics of the interface are defined, including all of the data that will be passed to the DLL model. The OE automated build process allows total description of the desired object to be made as items in an Excel spreadsheet. The resulting output is a complete set of VB code for the desired object (i.e., class file, control file, and form file).

In addition to OE, several other new system utilities were developed during the TSAT program. These include a controller to provide for control and convergence of iteration loops, and additional connector objects that provide capability for passing specified data between objects. Another very useful utility developed in the program is a Schematic Viewer, allowing real-time data display of simulation results in a schematic diagram of the physical model as execution proceeds.

3.3.2 Thermal Library Development

A number of the components required for TSAT were available as Fortran procedural subroutines in the existing P&W FITSS code, which had been developed to model thermal management systems. To convert these components to callable objects, several modifications were necessary. The first step was definition of a calling convention, or protocol, to define which variables would be passed and the format for passing variables. As originally written, component models in this code are called with reference to the location of the component in the integrated system, and the variables corresponding to each location are passed in named common blocks. Figure 7 shows an example of the interface entry for a heat exchanger (HEX01C) model. The argument variables show the location of the two inlet streams (IN1, IN2), the outlet streams (NX1, NX2), the numbered occurrence of the component (IPAN), and a defined table of physical properties (ITAB).

```
SUBROUTINE HEX01C(IN1, IN2, NX1, NX2, IPAN, ITAB)
```

Figure 7. Interface Definition for Procedural Heat Exchanger Model

Figure 8 shows how the same model was modified to provide an interface for a callable object as a DLL. The argument list now consists of several data arrays, passing the actual values of the input and output variables. In this case, ARR_N, GEOM_N, and PANEL_N include input variables and geometry, and IPOINT defines various integer switches and pointers. ARR_X includes general output data, and FIELD_X is the definition of distributed thermal characteristics for input to the WinTherm software.

```
SUBROUTINE HEX01 (ARR_N, ARR_X, GEOM_N, PANEL_N, FIELD_X, IPOINT)
```

Figure 8. Interface Definition for DLL Object Model of Heat Exchanger

To allow maximum use of the existing component, with minimum amount of rewriting code, an interface was written to act as a wrapper around the existing (legacy) subroutine. This wrapper, written in Fortran, translates the new input data arrays into common block arrays, compatible with the existing routine. In

this way, no common block data passing is used externally to the new DLL object, while all the internal common block structure is retained within the object. Most of these legacy components are composed of a number of subroutines, so retention of the original data structure significantly reduces the required effort to convert models. In addition to modifying the methods for data flow, many of the methods used in the original models were further developed to make the models more general in application. Also, access to fluid and material properties was modified to make the new component objects compatible with the new engineering utilities, also being developed during this program, and with the existing and new system utilities.

For those components that were not available from existing models, new models were written as part of TSAT. These new models were written either in VB or Fortran, depending on their complexity and nature. Simple models, such as splitters, were written in VB, while components that are more complex were modeled in Fortran with VB front ends.

To perform analysis of the processes occurring within a physical component, the thermophysical properties of the fluid streams flowing through the component must be known. These properties include the thermodynamic and transport properties of the fluids at their local state conditions, defined by any two thermodynamic properties (e.g., temperature and pressure). The property methods available in the prior FITSS code were not applicable for generalized use in an object-based environment, and therefore, were modified to provide a more generalized fluid property model. The resulting model retrieves property data from a database library, which is in turn based on standard references for various types of fluids. This model provides property data over the full range of state regions (i.e., solid, liquid, vapor), and handles specified mixtures. This model, and other engineering utilities, are developed as DLL objects, and are called from within VB objects or other Fortran DLLs. They have not been provided with graphical front ends, although they could be if desired. Engineering utilities, in general, provide information to or processes for component models, but are not called as standalone models.

3.3.3 WinTherm/RadTherm Interface

WinTherm and RadTherm are thermal analysis codes, developed by ThermoAnalytics, for modeling steady state and transient surface temperatures and infrared radiation (IR) of complex three-dimensional shapes. They are widely used for evaluation of IR signature from Army weapon systems, such as tanks and other armored vehicles. They are also used in the automobile industry to model hot sections of the automobile such as exhaust systems.

In aerospace systems such as a Hypersonic Ramburner as shown in Figure 9, there is a requirement to model the temperature distribution in a detailed structural component as well as in the fluid system, which transfers the heat from and to the structural component. To determine structural integrity and expected life of the structure, models must be developed to determine both the thermal distribution and the stress distribution in the cooling system structure. In general, different computer programs perform each of these functions. Although attempts have been made to automate the transfer of data from one program to the other, generally the form of these solutions are specific to a particular problem, and require tailoring of the interface for each problem. For this reason an attempt was made to couple WinTherm/RadTherm with ME, to identify the technical challenges associated with such solution coupling, and to demonstrate a potential method for resolution of those challenges. WinTherm and RadTherm are executable standalone programs, and are structured to handle input and output data in a binary file format (.tdf). There is a technical challenge in providing a means to transfer data in array format from a component object to a program accepting only structured data files, without requiring significant reprogramming.

To model a Thermal Panel, shown as part of the ramburner structure in Figure 9, a panel heat exchanger component object, HEX01, was developed in TSAT. This model was originally written to represent a convectively cooled section of a burner or nozzle structure, with coolant in the wall channels and hot

combustion products flowing along one side of the external wall. HEX01 can easily be combined with other Model Engineer components to model an entire cooling system. It was desired to couple the HEX01 model with a RadTherm model of the same structure. The HEX01 object had been modified under TSAT (beginning with the HEX01C procedural subroutine) to be a COM-compliant object with an array argument interface. Included in the HEX01 output is an array incorporating all the thermal variables needed to provide input to RadTherm, including geometrical data, fluid temperatures and heat transfer coefficients on both the hot and cold sides of the cooled structure. To provide data transfer capability between HEX01 and RadTherm, a COM object was developed that could accept the data array from HEX01, translate the data into the binary format readable by RadTherm, and write a .tdf file for input to RadTherm. This interface object also accesses RadTherm, causing it to read the input file and begin execution.

This RadTherm interface object can both read and write to the .tdf file, allowing it store both input data from the HEX01 object, and output data from RadTherm. This approach made the process very general and could easily be modified to represent any of the TSAT library components.

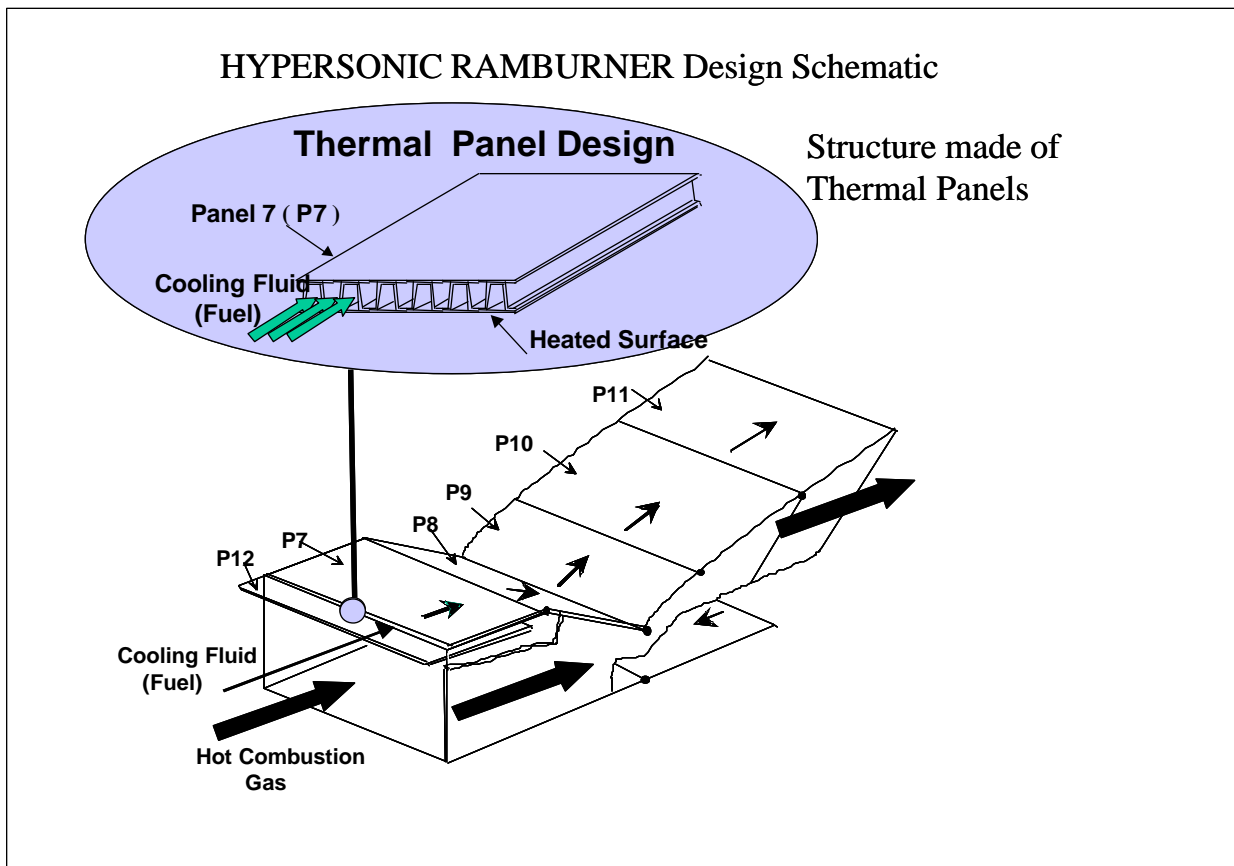


Figure 9. Cooled Burner Panel Used as Coupling Problem with WinTherm

Although not accomplished in this task, RadTherm has the ability to write its thermal data in file format, which can be then be used as input for NASTRAN, ANSYS, or other appropriate structural analysis code.

To demonstrate the data connection between the HEX01 and RadTherm objects, a sample case was set up with hot air (2800 F) flowing over the outer surface of a panel, and JP-8 fuel flowing within the cooling passages. A preliminary analysis was performed by Hex01, and the fluid characteristics were then

passed to RadTherm, which performed a detailed thermal analysis and predicted the surface radiation. The resultant output from RadTherm is shown in Figure 10. In this figure, the temperature profile over the heated surface is displayed as a color-coded grid, and is seen to vary from 404 F to 251 F. While the simple flat plate model used for demonstration is a trivial case, the same methods could be applied to more complex 3-dimensional models for thermal and radiation analysis of practical components.

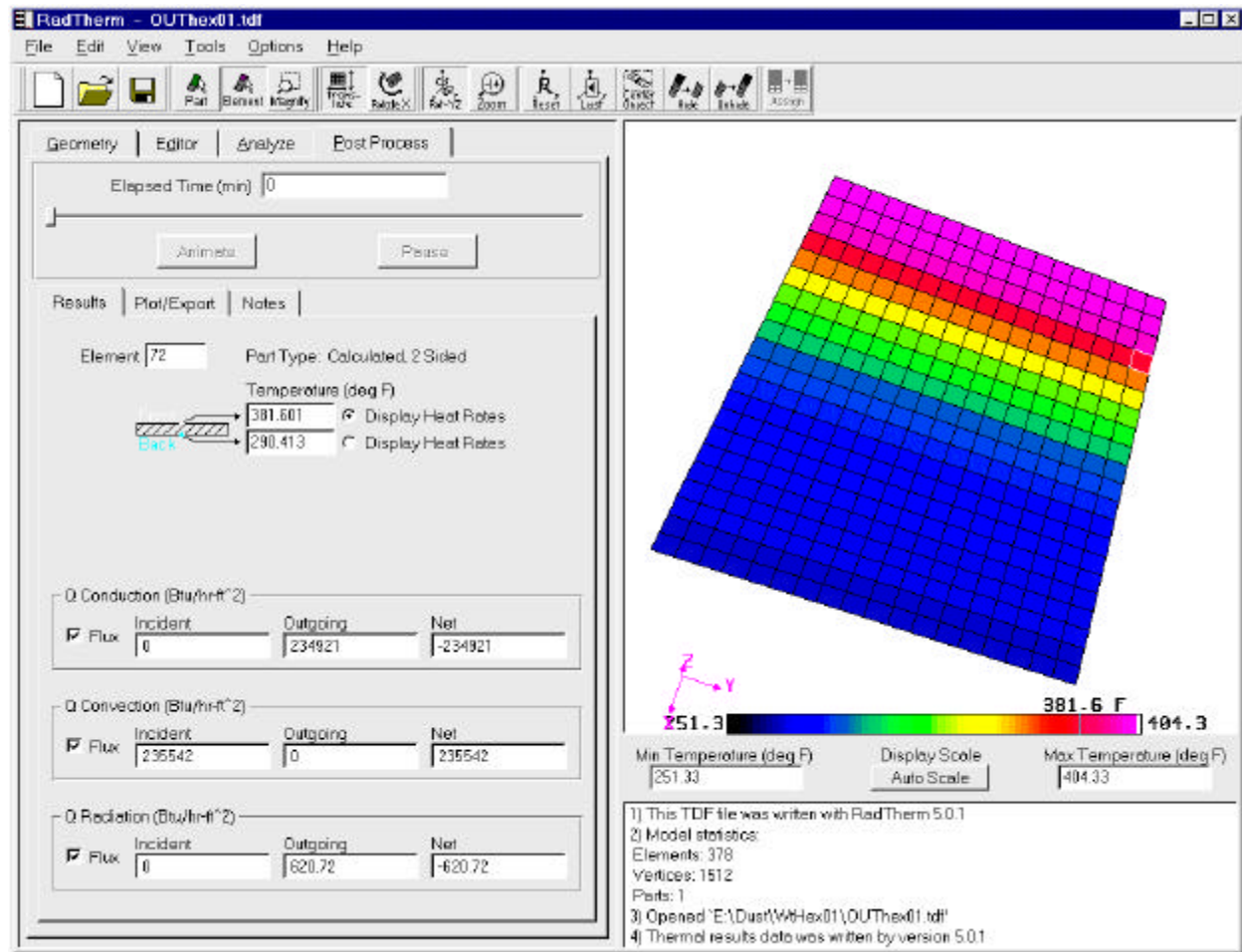


Figure 10. Output Display from RadTherm for Analysis of Heated Panel

4.0 Results

The TSAT program provided support for developing basic tools for building and analyzing integrated thermal systems, beginning with methods that had previously been developed. A set of object-based tools was developed for use under the Microsoft COM protocols. The results of the program can be best considered through examination of the developed tools and their capabilities, methods for application of those tools in a simple system, and validation of the methods when applied to a representative integrated system.

4.1 *The TSAT Toolbox Additions*

The TSAT tools have been developed to be consistent with the ME infrastructure, which is in turn based on Microsoft's Visual Studio/Visual Basic environment. In the VB COM-compliant environment, programming is done with a combination of graphical and textual entries. Commonly used methods are available as reusable code, represented by icons, that can be incorporated in a new program by drag-and-drop methods. These icons, representing ActiveX controls, are presented in VB in a window called the Toolbox. This is the same presentation method that is used for the new component models developed in TSAT, and the icons representing those new models are referred to as Tools. The tools representing the component models are ActiveX controls that can be dropped on a design form (i.e., the visual worksheet in Visual Basic) to graphically program an integrated system model.

A view of the Visual Basic development window is shown in Figure 11. The Toolbox is the small window on the left side of the figure, and the controls shown are a few of the large number available from the Microsoft Components Library. This design form represents the workspace, or environmental container, in which models are built. A control can be executed simply by dropping it on a form. Furthermore, the act of dropping multiple interactive controls on a form will result in generation of a file that will record the total structure of the form and all of its resident objects. This is the fundamental characteristic of a visual-programming environment that provides the capability for easy modeling of integrated systems. That is, the visual environment is more than a drawing program, it is also a powerful code-generation system, translating physical actions of the user into active code to represent the desired simulations.

Figure 12 shows the Toolbox with some of the ME and TSAT engineering tools added. These components, shown as icons in the toolbox, represent the various physical devices that were modeled in the program, and that are now available as ActiveX controls for simulation of integrated systems.

The tools developed in the TSAT program can be grouped into several classes of components. These include heat exchangers, turbomachinery components, fluid flow devices, and general engineering utilities. The component models developed in the TSAT program will now be described with regard to their scope, applications, and capabilities. All of the component models depend on the same engineering and system utilities, and are structured in a consistent format. As previously mentioned, some components are written totally in VB, and others are a combination of a VB front end, and a computational DLL written in Fortran. All components use the same fluid and material properties routines, and share a common set of passed properties, defining the fluid and its thermodynamic state in each stream passing through the various components in an integrated model.

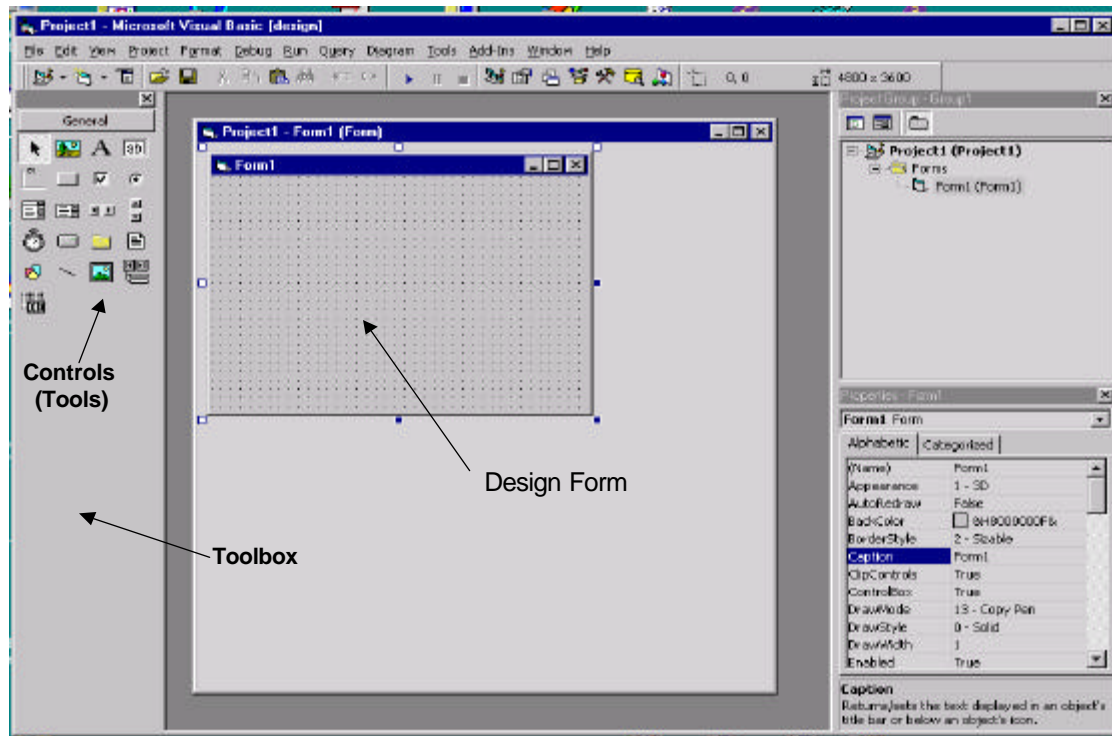


Figure 11. Visual Basic Development Window

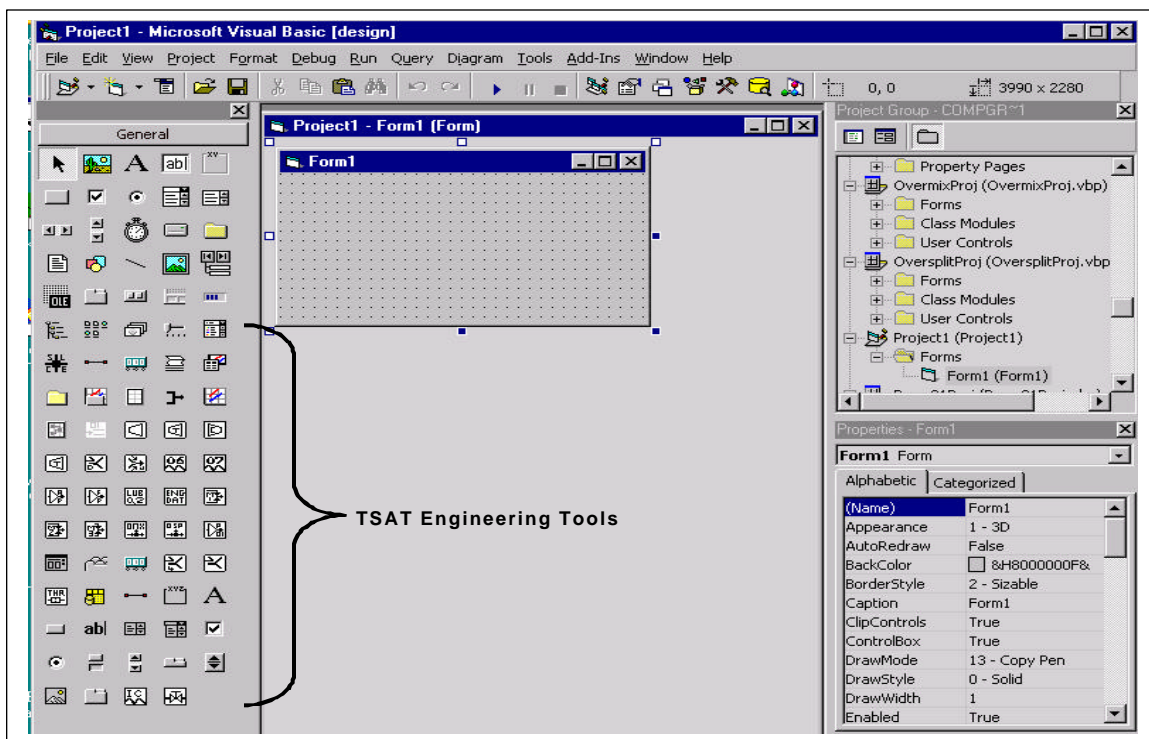


Figure 12. Visual Basic Toolbox with TSAT Engineering Tools Added

4.1.1 Heat Exchangers:

Three classes of discrete heat exchangers have been included in the TSAT library. These are all based on compact construction with generic sheet-fin passage arrangement. Specifically, these classes include a generic counterflow arrangement, a multipass cross-flow arrangement, and a single layer heat exchanger representing a cooled structural panel. Generic sheet-fin construction is loosely used here to represent the general arrangement of internal structures. Structure types available for design consideration include true plate and fin construction, and machined passages in laminated layers. When conventional fin-plate arrangement is used, the available fin types include all those evaluated and included in the Kays and London database.⁸ In addition, all fin geometry in the database can be scaled to desired dimensions and structural design details. If machined passages are desired, selected passage shapes can be circular, or rectangular with rounded corners. All dimensional details, including all internal and external design details, are variable and can be set by the user. This includes passage type, size, spacing, layer sizes, number of layers, and external dimensions. Materials are user-selected from a materials property library. The following list outlines the characteristics and capabilities of the available heat exchanger models.

Compact Counterflow Heat Exchangers

- Full counterflow or Z-path arrangement
- Capability to add a buffer layer between hot and cold layers, to provide safe separation between combustible fluids and hot air
- Fouling factor definition, describing effect of accumulated coatings on heat transfer surfaces
- Analyze heat transfer with one-dimensional finite difference methods
- Extended surface (fin) model treats second dimension
- Real fluid properties available at each calculation step
- Treats high Mach number internal flows with real-fluid influence coefficient methods
- Allow temperature or enthalpy as the as starting condition
- Evaluates heat exchanger core and manifold weight

Compact Multipass Cross-flow Heat Exchangers

- Analyze heat transfer with two-dimensional finite difference methods
- Extended surface model treats third dimension
- Capability to add a buffer layer between hot and cold layers, to provide safe separation between combustible fluids and hot air
- Allows multiple passes in cross-counterflow arrangement
- Mixed or unmixed flow in multiple pass arrangements
- Allows alternating or single direction multiple passes
- Will treat any fluid on either side of heat exchanger
- Real-fluid influence coefficient methods for high velocity flow
- Evaluates core and manifold weights

Single-Layer Cooled Panel

- Parallel flow single-layer heat exchanger
- Simulates convectively cooled panel
- Internal coolant is selectable
- External hot flow selectable fluid or combustion products
- External Mach number distribution and overall geometry can be defined by user
- Finite difference analysis of heat transfer and fluid streams

Intercooler (Defined Heat Load)

- Applies a user-defined heat load and pressure loss on a fluid stream
- Evaluates fluid exit conditions

4.1.2 Turbomachinery:

Rotating machinery components are necessary in systems for pumping fluids, and for power production. A centrifugal pump object is provided for pumping liquids, and a generalized compressor model for pumping gases. A generalized turbine model is provided for two purposes, for producing power and/or expansion of the stream as part of a Brayton refrigeration cycle.

Centrifugal Pump

- Use with liquids
- Scales performance from specified design point
- Can operate with performance map
- Uses real-fluid properties

Compressor

- Use with fluids in gaseous state
- Operate with defined efficiency or with performance maps
- Uses real-fluid properties
- Power and speed match with power source (e.g., turbine, motor)

Fan

- Low-pressure device
- Use with fluids in gaseous state
- Operate with defined efficiency or with performance maps
- Uses real-fluid properties

Turbine

- Use with fluids in gaseous state
- Operate with defined efficiency or with performance maps
- Uses real-fluid properties
- Power and speed match with power sink (e.g., compressor)

4.1.3 Fluid Flow Components:

Fluid flow components are the various physical devices that direct, connect, and control flow of the fluid streams among the various thermal and power components in an integrated system.

Duct

- Variable area
- Inlet and exit losses
- Exit static conditions for pressure matching

Control Valve

- Variable area valve with controllable pressure loss

Throttle valve

- Constant enthalpy expansion process

Flow Splitter (Controlled)

- Two-branched splitter with split control

Multibranch Splitter

- Splitter with up to four outlet branches
- Defined flow distribution among outlet streams
- All outlet streams are same fluid type
- Passes thermodynamic state of outlet streams

Recirculating Splitter

- Two-branched splitter with logic for exit of recirculating flow loop

Flow Divider

- Virtual flow divider for analysis of parallel arrays of components
- Allows analysis of one representative component out of multiple in array
- Example: Analyze one HEX out of fan duct heat exchanger parallel array

Mixer

- Two-branched mixer
- Both inlet streams must be same fluid type
- Evaluates thermodynamic state of exit stream

Multibranch Mixer

- Mixer with up to four inlet branches
- All inlet streams must be same fluid type
- Evaluates thermodynamic state of exit stream

Ejector-Mixer

- Two-branched mixer configured as ejector
- Same fluid type in both streams
- Calculates pumping effect and thermodynamic state at exit
- Example: Mixing of high and low momentum streams, such as high velocity bypass flow around a fan-duct heat exchanger (HEX) mixing with HEX discharge

Flow Joiner

- Virtual flow joiner for analysis of parallel arrays of components
- For use paired with Flow Divider

Recirculating Flow Joiner

- Two-branched mixer with logic for inlet of recirculating flow loop

Water Separator

- Use to separate condensed water vapor from incoming stream
- Performance based on device efficiency, either entered as constant or from map
- Use in conjunction with condensing heat exchanger or cooling turbine

4.1.4 General Engineering Utilities:

General engineering utilities are routines that provide support to the component object models, either in providing property data or in evaluating specific processes. Several such utilities were developed as part of the TSAT program, providing general thermophysical properties of fluids, psychrometric properties of air-water mixtures, properties of structural solids, and generalized isentropic gas dynamic processes of real fluids.

Generalized Fluid Properties

This utility, the Fluid Table Reader (FLTABL6) reads a table of thermodynamic properties based on an externally generated data file, and returns a complete set of thermodynamic and transport properties corresponding to the thermodynamic state point. The input thermodynamic state point can be defined by any of the following pairs of state conditions:

- Pressure (psia) and Temperature ($^{\circ}\text{R}$)
- Pressure (psia) and Enthalpy (Btu/lbm)
- Pressure (psia) and Entropy (Btu/ $^{\circ}\text{R}$ -lbm)
- Enthalpy (Btu/lbm) and Entropy (Btu/ $^{\circ}\text{R}$ -lbm)
- Pressure (psia) (For Saturation Properties)
- Temperature ($^{\circ}\text{R}$) (For Saturation Properties)

The utility is currently structured to handle data for five different classes of fluids, defined as:

- 1) Single nonreacting fluid (no saturation data)
- 2) Single nonreacting fluid (with detailed saturation data), including liquid, vapor, supercritical data
- 3) Mono-reacting fluid (single fluid with chemistry changes)
- 4) Bi-reacting fluid products (two fluids with chemistry changes)
- 5) Two-component gas mixture with a condensable component (e.g., water-air mixture)

Data file formats have been developed for each fluid class, and data generation procedures have been developed for a wide range of fluids of engineering interest, based on commercially available property software. Sources of data used in the fluid property library, or compatible for building additional fluid models, are available from the NASA CEA program,⁹ the NIST SUPERTRAPP program,¹⁰ and the NIST REFPROP program.¹¹

Generalized Material Properties

The properties of structural materials are necessary for any components in which thermal analysis is performed, such as heat exchangers, or any components with weight estimation. A general material routine, Material Table Reader (MATTABL) provides the required properties of selected materials as a function of temperature. The properties currently included in the routine are:

- Thermal Conductivity
- Density
- Specific heat
- Thermal Emissivity

One-Dimensional Gas Dynamics

This utility considers the local gas dynamics in a one-dimensional flow, and evaluates the relations between the following state properties:

- Gas flow rate (lbm/sec)
- Total temperature (R)
- Total pressure (psia)
- Total enthalpy (Btu/lbm)
- Mach number
- Static temperature (R)
- Static enthalpy (Btu/lbm)

Given various sets of known conditions, the utility determines the remaining state conditions. Analysis is based on real-fluid properties of the fluid in the stream.

Psychrometric Model for Air-Water Mixtures

The utility PSYCHRO calculates the psychrometric state of air-water mixtures based on NIST data for the saturation state of water, and ideal gas equation of state model for water vapor. Required input is either (Pressure, Temperature) or (Pressure, Enthalpy), plus one of the humidity variables (i.e., Specific humidity, Relative humidity, or Wet-bulb temperature). PSYCHRO returns the other humidity variables, plus the full set of thermodynamic information corresponding to the state condition.

4.2 Model Engineer System Utilities

These are general utilities that are used with the engineering tools for building simulations of integrated systems. These utilities also include objects that provide logistic support for data input/output, system model operability, and model editing.

4.2.1 Connectors:

Connectors manage the flow of information between connected objects, passing required property information between the represented components.

4.2.2 Controllers:

Controllers provide control simulation in the integrated system model, and manage the balancing of iteration loops to ensure convergence of such loops.

4.2.3 Data Generator:

The data generator controls the execution sequence of all the components in the integrated system simulation. It also provides a means for defining selected output data for saving in an output file, or displaying in graphical plots.

4.2.4 Schematic Viewer:

The schematic viewer provides an on-screen visual display of system data as a system model executes, providing an immediate overview of component performance and system interactions. This is a pop-up window with a schematic illustration of the physical system being modeled, either developed by the user or input from another source. System data is selected by the user and displayed in the schematic view, positioned appropriately relative to the individual components.

4.2.5 ReadXcells and WriteXcells

These objects allow reading from and writing data to a defined Excel spreadsheet during execution of a model. Specified columns and rows can be sequentially accessed during execution, allowing a series of operating conditions to be evaluating during a single execution of the model. Specific cell rows can be defined for input, output, or information; allowing input definition and corresponding output data collection on a single spreadsheet. ReadXcells is used for input, and WriteXcells for output.

4.2.6 DoLoop

The DoLoop provides the capability for sequentially applying the object execution sequence defined in the Data Generator. When used in conjunction with the ReadXcells and WriteXcells objects, the DoLoop object allows evaluation of a predefined series of operating points, with changing operating conditions and system parameters at each point, if so desired. This capability provides a very powerful process for evaluation of complex systems operating over a range of differing conditions.

4.2.7 PasteView ToPPT

PasteView ToPPT copies the image produced by the Schematic Viewer and pastes it into a PowerPoint presentation. The object automatically opens a new blank slide and then pastes the copied image. When used in conjunction with the DoLoop and ReadXcells objects, it allows saving all of the visual output from the Schematic Viewer in a presentation as the system evaluation proceeds.

4.2.8 EditProperties

The EditProperties object provides the capability for viewing and editing the data-passing connections between objects. It will display all data-paths for a selected object, and allow editing of the selected connection properties. This is a very valuable tool for identifying and correcting connection errors while building models, and for modifying component arrangements in existing models.

4.2.9 Log Watch

The Log Watch object provides capability for error trapping in the component objects, and collects generated error messages for review in the Error Log. In addition, the accompanying Info Log allows tracing the execution sequence and data passing history for later review and model checking.

4.3 *Object Engineer*

OE is a source code generator that aids in the building of new component models. This tool automates much of the process of building new component objects, and object interfaces for mixed language programming. For instance, to use a Fortran DLL in a graphical environment requires that a visual object interface be written to pass data and act as a calling routine. For a complex DLL, the Visual Basic interface object will itself be quite complex, providing a graphical connection, and translating all passed properties into the correct format for interfacing with the DLL. Writing such a graphical object can be performed manually, but it is somewhat of a tedious chore. Recognizing that the necessity for writing such objects would occur many times, Modelogics developed a tool to vastly simplify and speed up the process. With this tool, the data interface definitions are specified in an Excel spreadsheet, and OE processes the data and writes the Visual Basic files necessary to produce the graphical interface. With some minor additions in one of the files, the interface object is operational. This tool can also be used to write the basic code structure for objects that are stand-alone, without an associated DLL. In this case, the computational operations are added to the resulting Visual Basic files, written by OE. This capability has resulted in a tremendous reduction of effort and time associated with writing new component objects.

5.0 Application and Discussion of Program Results

With the TSAT tool set complete, it is now instructive to examine the procedures for applying the developed methods to specific problems. The Visual Basic workspace, in which the component objects were built, is also a very useful environment for building system models. As a first step, some individual component objects will be examined, to understand their general behavior.

5.1 Component Object Behavior

A visual object can be executed in a stand-alone mode in the VB workspace just by dropping it on a design form. Figure 13 shows the same workspace that was illustrated in Figure 12, but a counterflow heat exchanger object (labeled O6) has been dragged from the toolbox and dropped on the design form.

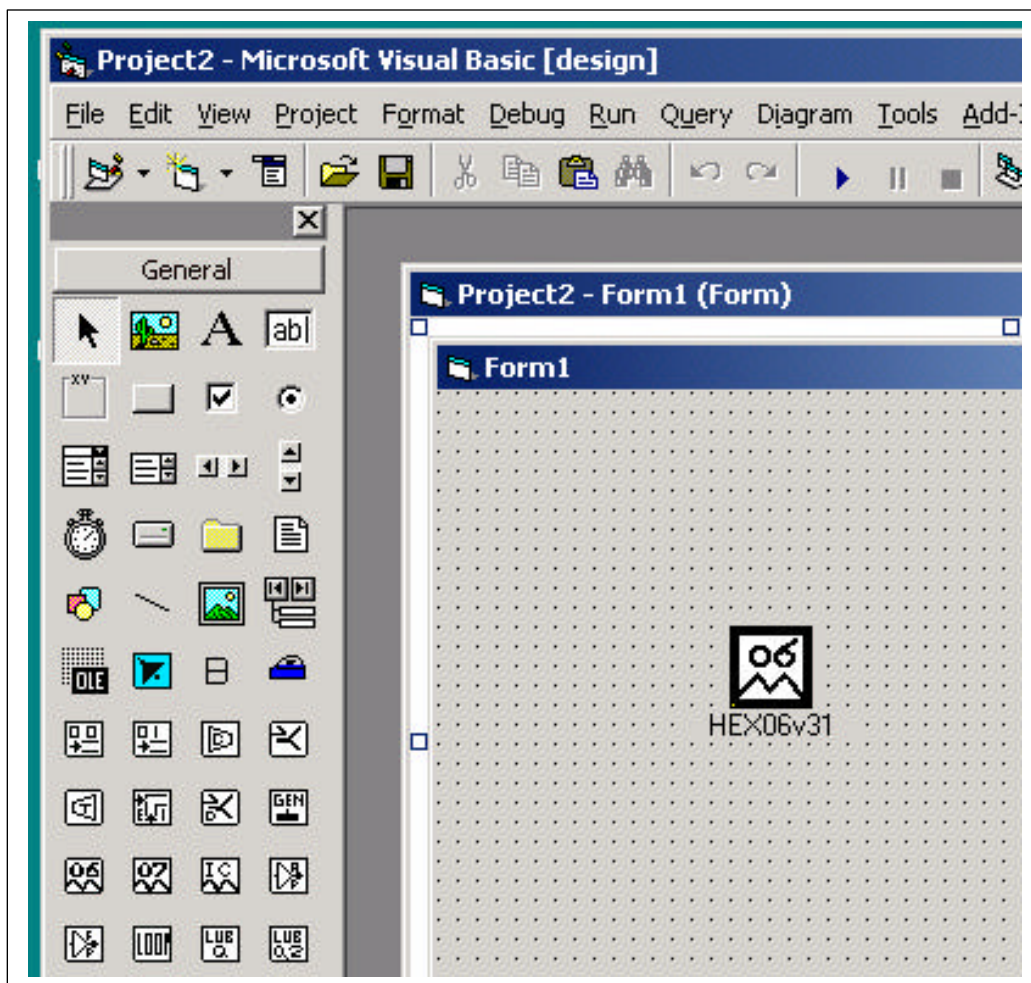


Figure 13. Counterflow Hex Object Dropped on Design Form

One of the characteristics of an object is that it has properties, consisting of all the variables that define the particular characteristics or instance of the object. The same object can be used many times in the same system model. Each time it is used is called an instance of the object, with each instance having

different properties. The properties can be viewed several different ways, but a convenient way is on the property page. This is a window that is defined while the object is being built, and the variables that the object designer wishes to display can be placed in this window. The object dropped on the form in Figure 13 represents a generic counterflow heat exchanger. This heat exchanger can be configured to represent a specific design by setting properties in the property page. The property page is opened for display by clicking on a menu that appears as part of the icon associated with the object. When the property page is opened, the object will run. That is, it will execute and calculate properties based on a default set of design parameters that have been built into the object. Figure 14 illustrates the first of a set of property pages associated with this particular heat exchanger object.

Figure 14. General Property Page for Counterflow HEX06v3

The first page is a general page, and displays several properties. The name is shown as HEX06v31, meaning that the name given to the object is HEX06v3 (i.e., Version 3), and the final '1' indicates that this is the first instance of the form on the page. When the property page is opened, the object executes, meaning that the code related to the object icon is loaded into memory and runs. For this object, there are two blocks of code, a Fortran DLL (hex06v3.dll) that does the detailed analysis, and a Visual Basic project (HEX06v3Proj.ocx) that generates the visual front-end object. In this particular case, the VB project file has been compiled into an ActiveX control. Objects can be executed from either the design project source code or as a compiled object, but are often left in the design mode for convenience while development is

in progress. Once run, the object remains in memory ready to run again if any properties change. Properties can be changed on the property page, and then pressing the Apply button will cause the object to execute and display the results of the property change. Pressing OK causes the object to run, and close the property page. Thus, an object can be run on a form in design mode by using the property pages. The object can also be executed by placing the form in run mode, to be demonstrated later.

The tabs at the top identify the property pages, and these can be labeled as desired. Shifting to the HxProps page, illustrated in Figure 15, some of the general design features of the heat exchanger are shown. Options are provided to input effectiveness for a simple cycle evaluation, or to do a detailed

The image shows a software window titled 'Heat Exchanger Property Page' with several tabs: 'General', 'Hx Props' (selected), 'HotSideFinProps', 'BufferProps', 'ColdSideFinProps', and 'Hx'. The 'Hx Props' tab contains the following properties and values:

Property	Value
CalcHeight	4.942
CalcNumColdLayers	12
CalcNumHotLayers	11
EffectivenessOption	PerformAnalysis
HexHeight	4.6
HexLength	8
HexWidth	5.75
InfoTrace	0
Input_Effectiveness	0.7
Option_Height/NumLayers	Enter Height
NumCalcSteps	10
NumberColdLayers	17
NumberHotLayers	16
Outer Layer	Cold Layer

Figure 15. Heat Exchanger Property Page Shows External Dimensions

performance analysis. An option is also provided to define the height of the heat exchanger, or to enter the number of layers. Since the details of the layers are defined (on another page), entering the number of layers will produce a particular height. If the desired height is input, then the number of layers will be calculated to approximately yield the desired height. The other dimensions shown represent the width of the heat exchanger in the no-flow direction, and the length in the direction of fluid flow. The number of calculation steps defines the breakup of the model into finite difference elements.

Figure 16 shows the HotSideFinProps page, containing hot-side fin design information. Properties that can be defined include the type of flow channels, type and details of fins, type of flow inlet and inlet dimension, and wall material. The ColdSideFinProps page has the same information for the cold flow side of the Hex. The BufferProps page offers an option to choose if the heat exchanger has a buffer layer. This is a barrier layer placed between adjacent hot and cold layers if combustible combinations are in use (e.g., fuel-air coolers). This design feature negatively impacts performance, but is sometimes used for safety.

General	Hx Props	HotSideFinProps	BufferProps	ColdSideFinProps	Hx
ChannelType_Hot		Fin Channel			
FinClass_Hot		Plain Fin			
FinsHeightHot (in)		0.1			
FinInputOption_Hot		Fin Data Table			
FinsThicknessHot (in)		0.004			
FinsPerInchHot (Fins/in)		22			
FoulFactorHot (hr-ft ² -F/Btu)		0			
FoulThickHot		0			
InputOptionHot		0			
FinNameLouver_Hot		Louvered Fin			
PassageConfigHot		1			
PassageHeightHot (in)		0			
InnerRadiusHot (in)		0			
OuterRadiusHot (in)		0			
PassageWidthHot (in)		0			
FinNamePlain_Hot		Plain Fin / 25			
PlateThicknessHot (in)		0.01			
FinNameStrip_Hot		Strip Fin / 1			
SurfRoughnessHot (in)		0.0001			
ZTurnLossFactorHot		3			
WallMaterialHot		1			
FinNameWavy_Hot		Wavy Fin / 1			
ZInletWidthHot		2.2			

Figure 16. Property Page Showing Hot-Side Properties

The HxPerf page, shown in Figure 17, contains boundary input conditions, and predicted performance data. As shown in the figure, the input data consists of default values that are included in the component model design. Since the object runs on opening, it is necessary to provide a set of initial default data so the initial execution doesn't fail.

On this page there are two columns of data, and some rows of data have corresponding left and right hand elements. For such rows, the left-hand column is the input data, and the right-hand column is the exit data. These items are called flow variables and represent stream information with inlet and exit values. Data items that are simply input values, or are single calculated values, do not appear as flow variables. For example, inlet and exit temperatures are shown, but calculated heat exchanger effectiveness only appears once. Also note that the fluids (FldNamCold, FldNamHot) entering the heat exchanger can be selected. In this case, air is flowing in both sides, and is shown as a property leaving the component. The name of the fluid in a stream is passed to each component with the thermodynamic properties, thereby ensuring that all subsequent components along the stream path know which fluid is flowing in the stream. Providing information in this way, at the beginning of each stream, avoids the necessity of setting the fluid for each component. The last two pages, not illustrated here, contain detailed weight and volume information for the heat exchanger core and associated manifolds.

HotSideFinProps	BufferProps	ColdSideFinProps	HxPerf	HxWeights	Finl
ChemistryCold (none)	0	XcoldOUT (none)	0		
DelPrCold	0.25				
HcoldIN (Btu/lb)	0	HcoldOUT (Btu/lb)	101.2439		
FlowcoldIN (lb/min)	90	FlowcoldOUT (lb/min)	90		
FldNamCold	Air	FldNamCold	Air		
PcoldIN (psia)	30	PcoldOUT (psia)	27.86415		
PstaticCold	0	PScoldOUT (psia)	27.52904		
TcoldIN (°F)	260	TcoldOUT (°F)	493.2472		
HeatCapRatio	0.4690676				
HexEffectiveness	0.5913994				
ChemistryHot	0	XhotOUT (none)	0		
DelPrHot	0.25				
HhotIN (Btu/lb)	0	HhotOUT (Btu/lb)	128.6273		
FlowhotIN (lb/min)	40	FlowhotOUT (lb/min)	40		
FldNamHot	Air	FldNamHot	Air		
PhotIN (psia)	60	PhotOUT (psia)	53.59484		
PstaticHot	0	PShotOUT (psia)	53.31333		
ThotIN (°F)	1100	ThotOUT (°F)	603.2245		
HexNTU	1.165751				

Figure 17. HxPerf Page Shows Performance Data

Another mode in which component objects can be executed is when integrated as part of a system. In this mode, component objects are sequentially executed, and data is passed in the desired sequence by means of connectors. The Data Generator object controls the sequence of operation, as defined by the user. These connectors, which are system utilities, do not represent physical pipes with fluid flow, but rather are virtual flow devices, passing selected data between connected components.

5.2 Integration of Multiple Components

To illustrate the process of integrating components, two counterflow heat exchangers are combined into a simple system. The same air-air heat exchanger, shown in the previous example, is combined with a second heat exchanger, in which JP-8 fuel is used to further cool the hot air exiting from the first heat exchanger. To execute an integrated system, the design form must be put in run mode, and a Data Generator object provided to control sequential execution of the components. Figure 18 illustrates the integrated system, with a connector passing data from the first heat exchanger on the left to the second heat exchanger on the right. The data generator is seen on the form, which is the control device for operation of the model. Figure 19 shows the Execution Order in the Property Page for the Data Generator.

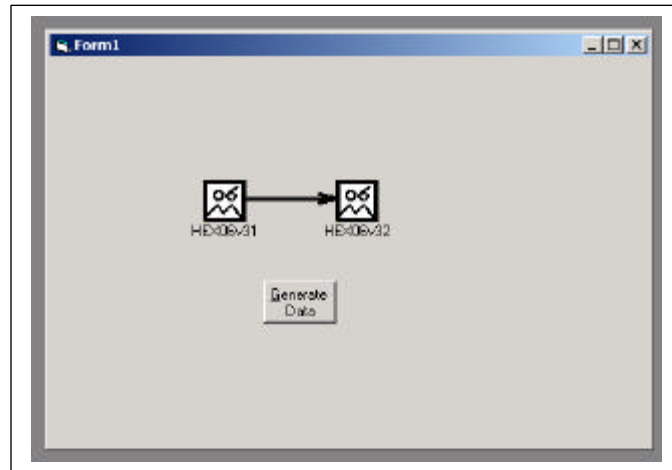


Figure 18. Simple Integration of Two Heat Exchangers

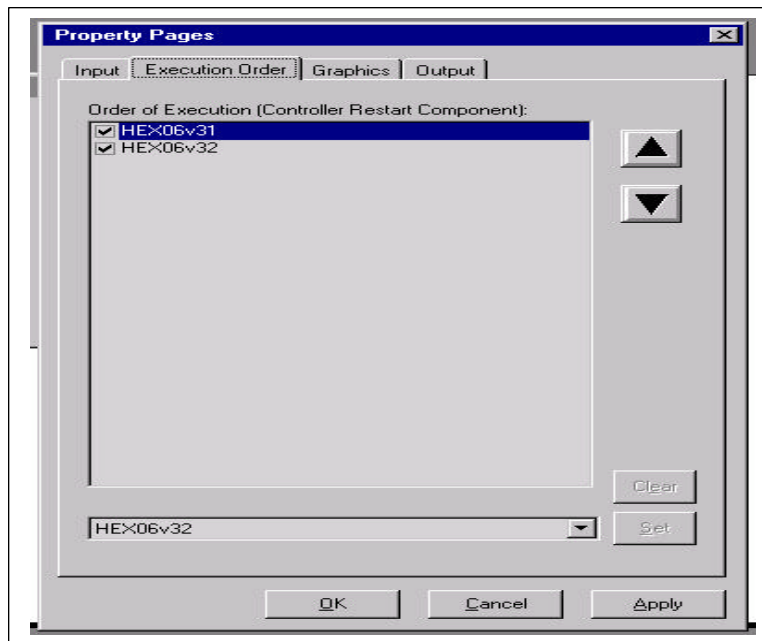


Figure 19. Execution Order is Defined in Data Generator

The performance data for the two heat exchangers is shown in Figures 20 and 21. Examination of these figures shows that the exit conditions of the hot stream from the first heat exchanger are the entry conditions into the second heat exchanger, just as the model intended. Although the connector is not a physical device, it is an object and has a property page.

HotSideFinProps	BufferProps	ColdSideFinProps	HxPerf	HxWeights	Fin
ChemistryCold (none)	0	XcoldOUT (none)	0		
DelPCold	0.25				
HcoldIN (Btu/lb)	0	HcoldOUT (Btu/lb)	101.2439		
FlowcoldIN (lb/min)	90	FlowcoldOUT (lb/min)	90		
RdNamCold	Air	RdNamCold	Air		
PcoldIN (psia)	30	PcoldOUT (psia)	27.98415		
PstaticCold	0	PScoldOUT (psia)	27.52904		
TcoldIN (°F)	260	TcoldOUT (°F)	493.2472		
HeatCapRatio	0.4630676				
HxEffectiveness	0.5913934				
ChemistryHot	0	XhotOUT (none)	0		
DelPHot	0.25				
HhotIN (Btu/lb)	0	HhotOUT (Btu/lb)	128.6273		
FlowhotIN (lb/min)	40	FlowhotOUT (lb/min)	40		
RdNamHot	Air	RdNamHot	Air		
PhotIN (psia)	60	PhotOUT (psia)	53.59484		
PstaticHot	0	PShotOUT (psia)	53.31333		
ThotIN (°F)	1100	ThotOUT (°F)	603.2245		
HxNTU	1.165751				

Figure 20. Performance of First Heat Exchanger, Cooled With Air

Hx Props	HotSideFinProps	BufferProps	ColdSideFinProps	HxPerf	HxW
ChemistryCold (none)	0	XcoldOUT (none)	0		
DelPCold	0.25				
HcoldIN (Btu/lb)	0	HcoldOUT (Btu/lb)	710.6994		
FlowcoldIN (lb/min)	60	FlowcoldOUT (lb/min)	60		
RdNamCold	JP-8 Fuel	RdNamCold	JP-8 Fuel		
PcoldIN (psia)	200	PcoldOUT (psia)	199.9651		
PstaticCold	0	PScoldOUT (psia)	199.9651		
TcoldIN (°F)	100	TcoldOUT (°F)	201.265		
HeatCapRatio	0.3172434				
HxEffectiveness	0.6774845				
ChemistryHot	0	XhotOUT (none)	0		
DelPHot	0.25				
HhotIN (Btu/lb)	128.6273	HhotOUT (Btu/lb)	44.75302		
FlowhotIN (lb/min)	40	FlowhotOUT (lb/min)	40		
RdNamHot	Air	RdNamHot	Air		
PhotIN (psia)	53.59484	PhotOUT (psia)	48.31588		
PstaticHot	53.31333	PShotOUT (psia)	48.70616		
ThotIN (°F)	603.2245	ThotOUT (°F)	262.2977		
HxNTU	1.405135				

Figure 21. Performance of Second Heat Exchanger, Cooled with Fuel

Examination of the Connector property page, shown in Figure 22, shows the alignment of data flow from one component to the next. In this case it is apparent which data items flow from the first heat exchanger to the second. This property page appears as connections are being arranged, and allows checking to ensure that the data connections are correct, and that data is flowing in the desired direction.

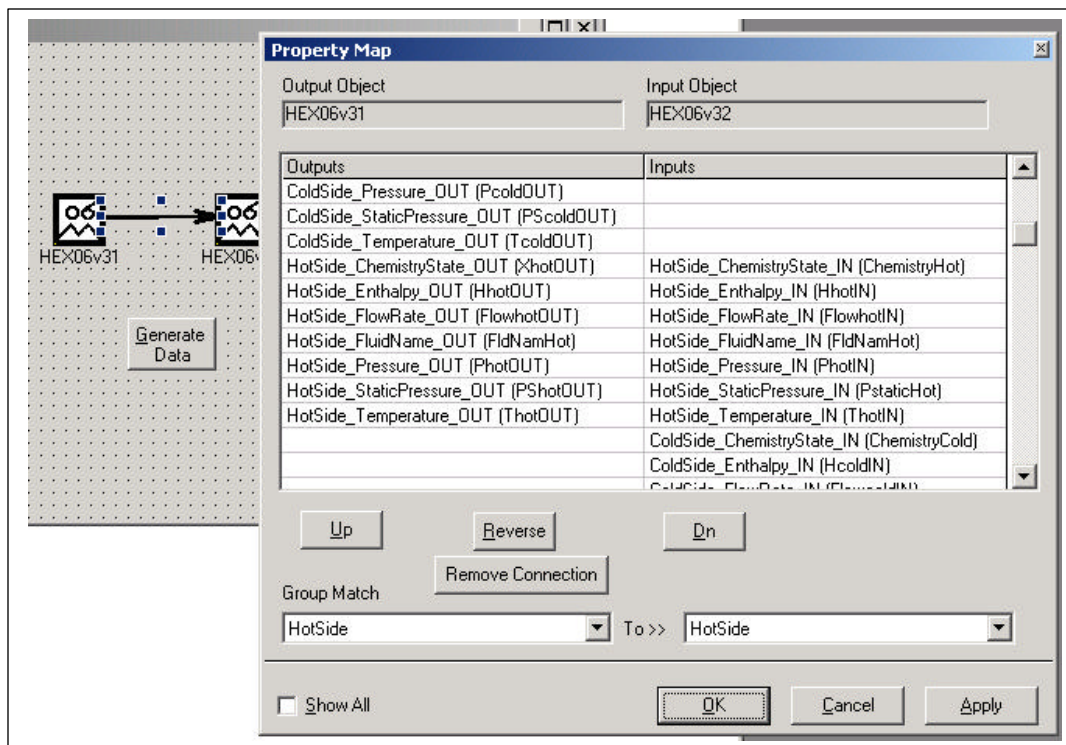


Figure 22. Property Page for Connector

Results of the analysis of a component or an integrated system can be examined by displaying the performance data on the corresponding Property Page. There is capability provided in the Property Page object for each component to write all of the component properties to a file for later viewing, but this is not very convenient if several components are involved. Several methods are provided in the ME infrastructure for viewing analytical results. The available methods for capturing output include the following:

- Write properties to output file from Property Page
- Write selected output properties to Excel file automatically from Data Generator
- Write selected properties to Excel file from WrieXcells
- Produce plotted graph directly on model form as execution proceeds
- Display property data in Schematic Viewer, showing schematic diagram of physical system

The use of a Schematic Viewer will be shown in the next section, as it is used in simulating a complex integrated system.

5.3 Integration of a Complex System

To examine the capabilities for modeling integrated complex systems, it is instructive to consider an application that challenges the system capabilities. To provide such a challenge, an aircraft with highly integrated subsystems is examined over a range of flight conditions. The selected flight system is a single-engine advanced strike fighter aircraft with high-level system integration. The vehicle ECS, which provides cooling for avionics and the cockpit, is integrated with the engine, using compressor bleed as a refrigerant in a reverse Brayton cycle, and fan air as the primary heat sink. The fuel system is integrated with the engine and vehicle thermal management systems, providing cooling for engine lubrication, vehicle hydraulic system, and generators. The system architecture is shown schematically in Figure 23, which is also the schematic diagram used in the Schematic Viewer in the system simulation.

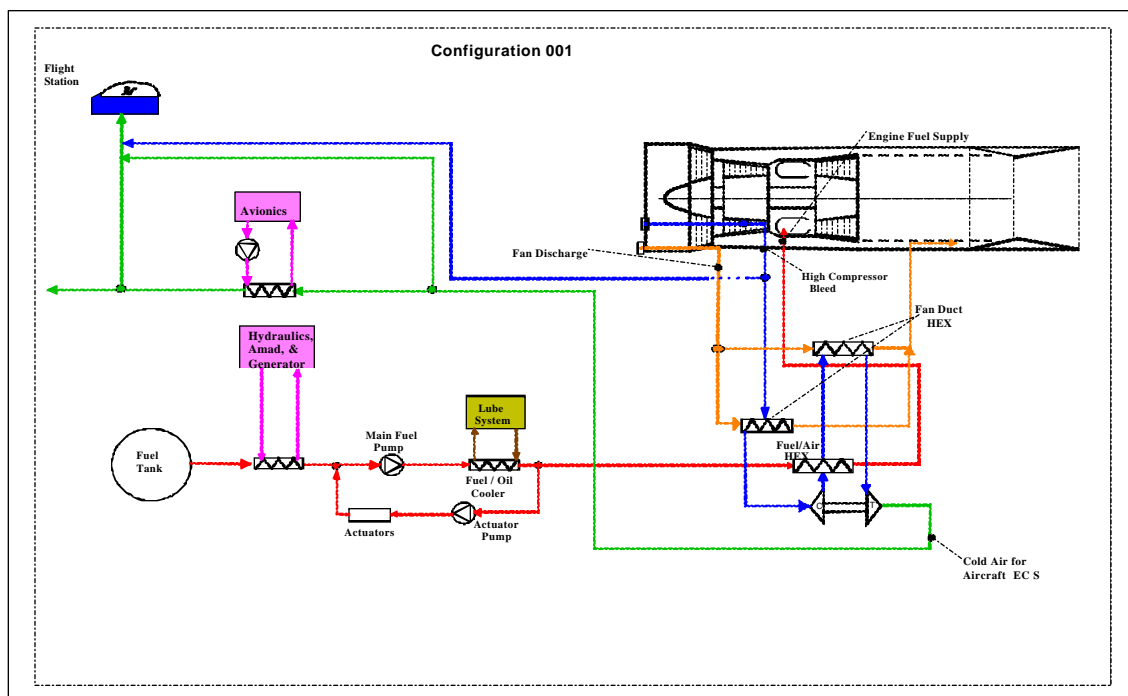


Figure 23. Architecture for Vehicle-Integrated Thermal Control System

A model has been integrated to represent the thermal control system shown in Figure 23. For use in this model, an engine object was developed from a P&W proprietary parametric engine model to provide engine performance information to the rest of the integrated system. The resulting model of this integrated system is shown in Figure 24. This model contains components that represent all of the details of the physical system. In addition to the flow components, controllers have been added to manage the iteration and convergence logic associated with several parts of the system. Specifically, a controller (TMS Loop) has been used to balance the iteration required to analyze the recirculation flow loop associated with the thermal management system for the engine, shown in the lower left portion of the model diagram. Controllers have also been added to control speed and power balance of the cooling turbomachinery (Comp Ctrl, Turb Area). Also, controllers are used to provide the desired environmental conditions. These control the various flows and flow splits to govern the cooled air temperatures. All of the physical components and flow distribution systems shown in Figure 23 are represented as model components in Figure 24. In addition, a splice object is used in a number of locations in the system. This is not a physical device, but rather a method of arranging the connector lines in the model diagram. This object simply passes data, with no operation on the passed data.

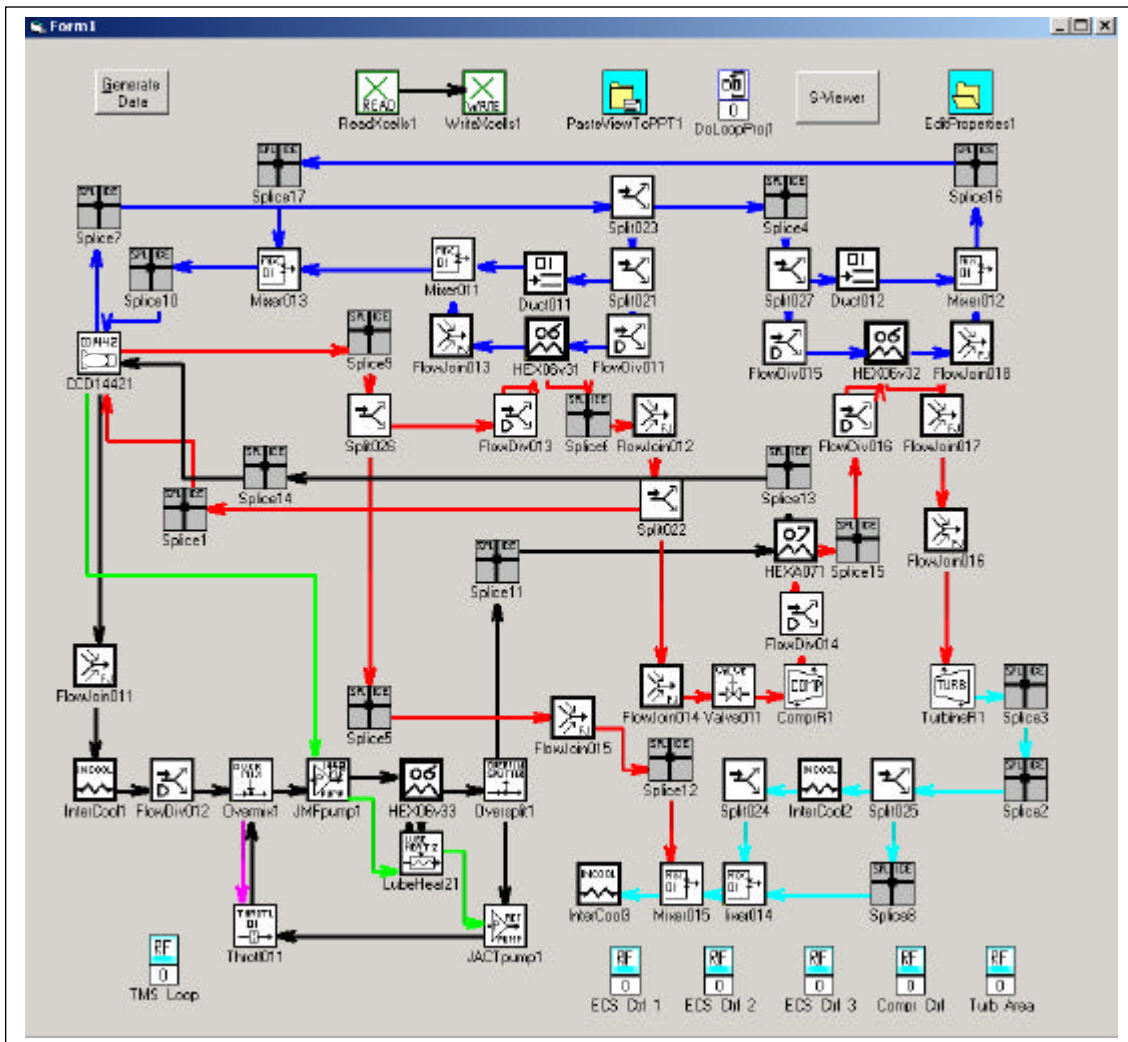


Figure 24. TSAT Model of a Vehicle Integrated Thermal Control System

Figure 24 is a screen capture of the integrated model in the Run mode. The color coding of the connections in Figure 24 are defined as follows: Engine high bleed - red, Fan air - blue, Fuel - black, ECS coolant - aqua, and Engine data bus - green. As the model is assembled, component objects are dragged from the toolbox and dropped on the design form. When the system model is completed, all of the relevant program code is assembled and referenced, but not yet ready to be executed. To allow execution, the code must be converted to a machine-readable form, either by compiling or by an intermediate interpretive step, as used by Visual Basic. When the Run command is given in the design environment, the program code representing each component object is converted into an intermediate language (p-code), which can then be read by an interpreter and executed. It is in this Run Mode that the model is executed, and appropriate data is passed between components.

It should be noted in the illustration of this model integration, that the individual component objects representing physical components vary greatly in the degree of fidelity, but are still able to simulate components and pass property data within the integrated system. In fact, with this methodology each component object may be modeled to any desired level of fidelity and may still be integrated into the system and communicate with other components. The only requirements for this functionality are that the

object must be COM compliant, and its interface properties must be exposed. This capability to integrate widely diverse component models is unique to modeling in an open object oriented environment.

Figure 25 illustrates the beginning of the execution order for this integrated system, as it appears in the Data Generator. Since this model contains more than 60 components, they cannot all be seen simultaneously in this view of the execution order. However, the list is scrollable, allowing sequential viewing of all the execution sequence. Also, the data generator allows changing the sequence as desired, and disabling selected component execution during model development and debugging.

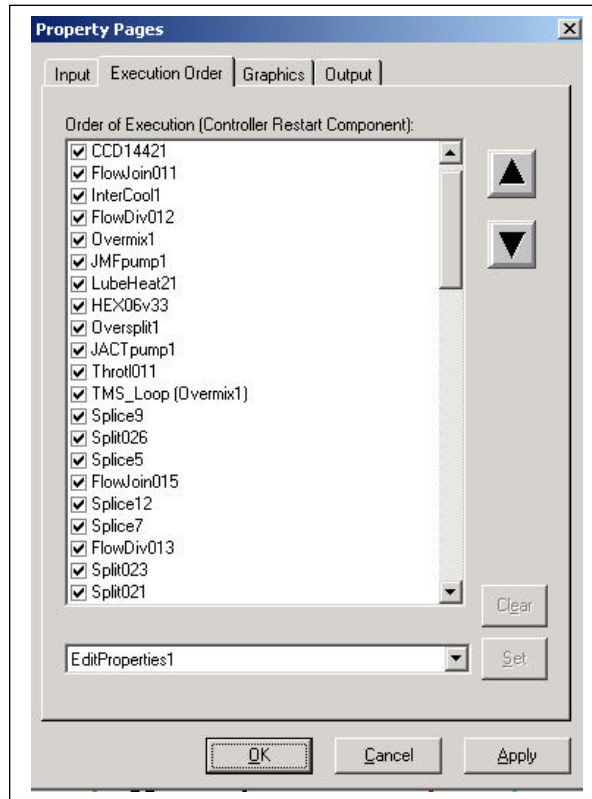


Figure 25. Execution Order of Integrated Thermal Model

Figure 26 shows the output display in the Schematic Viewer for a selected operating condition, high altitude subsonic cruise (40 K/0.9 M). The Schematic Viewer provides capability to display any desired property in the integrated system. In this schematic the flow connections are colored as follows: Engine bleed - blue, Fan air - orange, Fuel - red, and cold ECS air - green. The data displayed in Figure 26 comprise the most significant thermodynamic states in the system. The effect of any component modification, or change in operating condition, is readily seen in the viewer. To evaluate the effect of any change on the system, the desired change is made in the property page for the relevant component, and the system execution is repeated by executing an analysis step with the Data Generator. As this occurs, the simulation continues to run in memory. As each change is made, the relevant Schematic Viewer display is printed or saved to a file, providing a record of the analytical exercise. From these resulting displays, the sensitivities of the integrated system to design changes can be readily evaluated.

This method of visual system display provides an immediate view of the performance of an integrated system at a single operating point. If critical data items are selected for display in the viewer, the ability of the system to satisfy critical design requirements is instantly evaluated, and design changes can be interactively assessed. This provides the engineer with a tool that allows the design of a system, right

5.4 Development of a System SuperObject

One characteristic of object-based programming is the ability to build a subsystem with component objects, and then compile the total subsystem into a single object, with a defined set of properties. This capability can be very useful at times, allowing development of a subsystem object that can then be studied individually, or merged into a larger system. Such integrated objects, when constructed by Modelogics, are called SuperObjects, and can be packaged for easy use by the casual user.

A system of this type was developed as part of the TSAT program, for the purpose of demonstrating the interactive modeling of electrical power devices. A simple power system, based on a superconducting generator, was modeled to evaluate the potential benefits as compared to a conventional generator. The superconducting generator must be cooled to a specified low temperature to take advantage of the benefit of electrical superconductivity of the windings. The electrical resistance rises rapidly with temperature above the specified condition. A cryogenic refrigeration device (cryoCooler object) is required to maintain the necessary low temperature, and this device itself uses power. The system integration question is this: What combination of generator type and cooling heat sink (ram air or engine bleed air) provides the lightest power supply for a specified load capability?

A SuperObject model was developed, with appropriate option choices and output display, and was used to perform the desired trade study. The screen display for this model is shown in Figure 27.

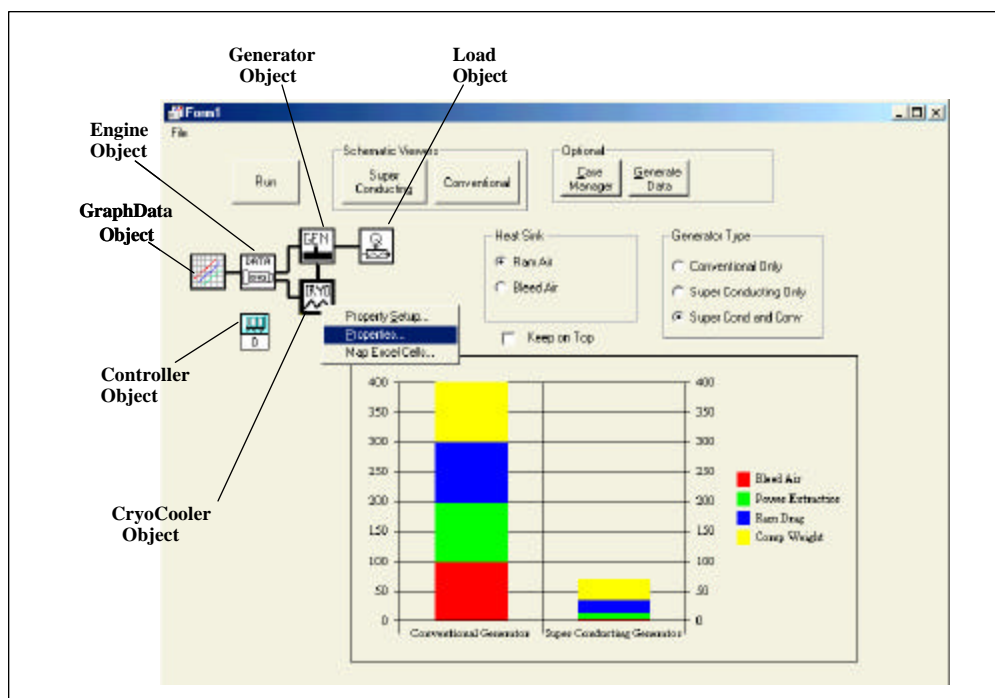


Figure 27. SuperObject Model of a Cryogenic Superconducting Power Supply

The resulting model presents a very friendly user interface, and is easy to use with very little training. This type of model is very useful for studies of commonly used systems, where options can be provided in the model to allow parametric variations in the system and operating conditions. Development of such models by practiced users of ME, for use by more casual users, can provide broad access to the benefits of the methodology.

6.0 Conclusions

Over the course of this program, TSAT, a comprehensive approach has been developed for the design, simulation, and analysis of integrated systems for thermal management and power generation. Basing the approach on compliance with Microsoft's COM, the open architecture underlying the integration of Microsoft's Windows applications, the resulting tools and methods developed under TSAT have been found to provide a very powerful approach for rapid prototyping and analysis of complex integrated systems. Furthermore, the resulting modeling methods have been shown to be applicable to a much broader range of applications than the initial focus on thermal system analysis. Demonstrations of the methodology have now been performed with total integration of air vehicle systems, with very promising results. The TSAT methods have now been applied to several other programs, including 4 government funded studies, and 2 commercial programs. The power and flexibility of the approach has raised significant industry interest, particularly for application in design and integration of advanced flight systems. In addition to pure thermal applications, the methods have been shown to also be applicable to other engineering disciplines, such as electrical power systems, in which electrical and thermal characteristics may be highly interactive.

Although the tools developed in the program provide a broad capability, the status of the tasks at the end of the current program should be considered as a work-in-progress rather than a final act. As the methods have been applied to new problems, new component requirements have been identified. In addition, the environmental architecture is in progress of further development (i.e., the Microsoft.NET initiative) and refinement to provide a more powerful environment for distributed computing. The methods of the TSAT program are readily capable of being merged into this future environment for open distributed computing. This program has established a strong beginning for a growing simulation methodology for future applications. The future growth of this methodology depends on its acceptance by industry, which will strongly depend on the exposure of these new capabilities and their successful application to solve challenging problems.

With the strong desire, both in the government and industry, to reduce cost of development of new aircraft systems, the ability to simulate and characterize the total flight vehicle during the early design phase will produce very positive benefits. The technology drive for the future should be a push to allow collaborative integrated modeling of total flight systems, including airframe, propulsion, and all supporting subsystems, at levels of definition sufficient to allow design optimization at the vehicle mission level. The methods initiated in the TSAT program provide a powerful indication of the direction to achieve such lofty goals, and the potential that might be achieved.

7.0 References

- 1) Carter, Howard (Nick) III, "Joint Strike Fighter (JSF)/Integration Subsystem Technology (J/IST) Demonstration Program," Volume 1: Executive Summary, AFRL-VA - WP-TR-2001-3028, 15 August 1995 - 30 November 2000.
- 2) Francesco Balena, Programming Microsoft Visual Basic 6.0, Microsoft Press, 1999, pp. 813-820.
- 3) "Vehicle Integrated Thermal Management Analysis Code Version 4.0, VITMAC User's Guide," Science Applications International Corporation (SAIC), December 1996.
- 4) V.J. Van Griethuysen, M.R. Glickstein, and E.S. Hodge, "Integrating Subsystem And Engine System Assessments," Proceedings of IGTI Conference, ASME Turbo Expo 2001, Paper 2001-GT-0452, June 4-7, 2001, New Orleans.
- 5) Howard (Nick) Carter III, Kurt Rupe, Robert Mattes, and Douglas Wiese, "Subsystem Utility Integration Technology (SUIT), Phase II," ADB226001, May 01, 1997.
- 6) Duncan J. Weber, Scott W. Van Horn, and Bernard P. Hill, "Vehicle Integration Technology Planning Study (VITPS)," AFWL-VI-WL-TR-97-3036, ADB233393, February 1997.
- 7) V.J. Van Griethuysen et al., Developments in High Speed-Vehicle Propulsion Systems, High-Speed Flight Thermal Management, Volume 165, Progress in Astronautics and Aeronautics, AIAA, 1996.
- 8) W.M. Kays, A.L. London, and David Bernhard, Compact Heat Exchangers, Krieger Publishing Company, January 1998.
- 9) Bonnie McBride and Sanford Gordon, "Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications," NASA Reference Publication 1311, June 1996, Lewis Research Center.
- 10) Marcia L. Huber, "NIST Thermophysical Properties of Hydrocarbon Mixtures Database (SUPERTRAPP), Version 2.0," U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, July 1998.
- 11) M.O. McLinden, E.W. Lemmon, S.A. Klein, and A.P. Peskin, "NIST Thermodynamic and Transport Properties of Refrigerants and Refrigerant Mixtures Database: Version 6.0 (REFPROP)," U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, July 1998.

8.0 LIST OF ACRONYMS

<u>ACRONYM</u>	<u>DESCRIPTION</u>
APU	auxiliary power unit
ASME	American Society of Mechanical Engineers
CEA	Chemical Equilibrium with Applications
CFD	computational fluid dynamics
CHER	catalytic heat exchanger reactor
COM	Component Object Model
DLL	dynamic link libraries
ECS	environmental control system
EPU	emergency power unit
FITSS	Fuel and Integrated Thermal System Simulator
FLTABL6	Fluid Table Reader
HEX	heat exchangers
IR	infrared radiation
J/IST	Joint Strike Fighter /Integrated Subsystem Technology
MATTABL	Material Table Reader
ME	Model Engineer
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
OE	Object Engineer
OLE	Object Linking and Embedding
P&W	Pratt & Whitney
SUIT	Subsystem Utility Integration Technology
T/EMM	Thermal/Energy Management Module
TSAT	Thermal Systems Analysis Tool
VB	Visual Basic
VITMAC	Vehicle Integrated Thermal Management Analysis Code
VITPS	Vehicle Integration Technology Planning Study